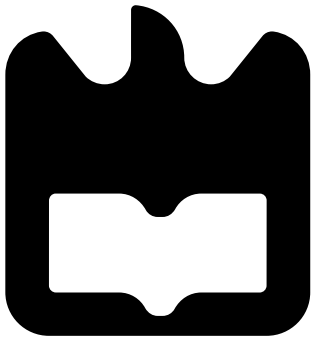
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática
2017

José Manuel Nunes de Brito Rosa

Modelação Comportamental da Camada Física de NB-IoT em Downlink
Behavioral Modeling of the NB-IoT Downlink Physical Layer



José Manuel Nunes de Brito Rosa

Behavioral Modeling of the NB-IoT Downlink Physical Layer

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Eletró nica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Arnaldo Silva Rodrigues de Oliveira, Professor do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Pedro Miguel da Silva Cabral
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar da Universidade de Aveiro (orientador)

Professora Doutora Maria do Carmo Raposo de Medeiros
Professora Associada da Universidade de Coimbra (arguente)

agradecimentos / acknowledgements

Primeiramente, um especial obrigado à minha mãe e avós por todo o apoio que me deram, sem eles não teria sido possível.
Ao professor Arnaldo Oliveira pela orientação para a concretização do projecto.
Ao Departamento de Engenharia Eletrónica e Telecomunicações da Universidade de Aveiro pela excelência no ensino.
Ao Instituto de Telecomunicações pelas excelentes condições de trabalho assim como equipa técnica pronta e competente.
Por fim não posso esquecer os meus amigos por todo o apoio prestado e por terem feito estes 6 anos inesquecíveis.
A todos, muito obrigado.

Palavras-Chave SDR, IoT, NB-IoT, simulador, downlink

Resumo

O paradigma da Internet of Things (IoT) define uma rede interligada de dispositivos que permite o surgimento de novas formas de interação entre utilizadores e dispositivos. O constante crescimento destas redes assim como a crescente demanda por uma fiabilidade maior, bit rates mais baixos e circulação massiva de informação insurgiu o aparecimento de novas tecnologias tais como as Low Power Wide Area Networks (LPWAN).
Em Junho de 2016 a 3GPP, o órgão responsável pelo LTE, lançou um novo standard para bandas licenciadas o Narrowband (NB) -IoT. O NB-IoT foi desenhado com base nos mesmos princípios que as outras LPWAN com o acréscimo de uma maior cobertura assim como uma mais fácil integração em sistemas celulares existentes.
Nesta dissertação aborda-se um estudo á sua camada física (PHY Layer) juntamente com uma implementação comportamental open source em Matlab das cadeias de transmissão e receção em downlink. O projecto modela e executa transmissões em downlink (eNodeB) e sua receçcão (terminal) tanto em ambiente simulado como integrado com um dispostivo de software defined radio, USRP, para validação laboratorial.
Os resultados obtidos tanto de simulação como co-simulação são apresentados avaliando a qualidade de transmissão assim como o comportamento do equalizador Zero Forcing implementado.

Key words SDR, IoT, NB-IoT, simulator, downlink

Abstract

The Internet of Things (IoT) paradigm defines a fully connected network of devices enabling new forms of interaction between users and devices. The constant growth of these networks, as well as an increasing demand for more reliable, low bit rate and massive communication data flows lead to the emergence of new technologies and set of standards, such as, the Low Power Wide Area Networks (LPWAN).

In June 2016, 3GPP, the consortium responsible for LTE development and standardization, released a new licensed band based standard, named Narrow Band (NB) IoT. NB-IoT was designed based on the same principles of other LPWAN standards, providing better coverage and additionally an easier integration on existing cellular systems.

In this dissertation a study on the NB-IoT Physical Layer is presented along with an open source behavioral implementation in MATLAB of the downlink transmission and reception chains.

The system generates and recovers one radio frame successfully performings procedures such as MIB and SIB1-N extracting along with scheduling and recovering data scheduled through control channels by higher layer paramenters. The project models and executes the downlink transmission (eNodeB) and reception (terminal) PHYs either in a pure simulation environment using different channel models, as well as integrated with an USRP software defined radio device for co-simulation.

The simulation and co-simulation results are presented evaluating the transmission's quality and performance of the implemented Zero Forcing equalizer.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP 3rd Generation Partnership Project

5G 5^{th} Generation

ADC Analog to Digital Converter

AWGN Additive White Gaussian Noise

BCCH Broadcast Channel

BER Bit Error Rate

CAZAC Constant Amplitude Zero Autocorrelation

CFO Carrier Frequency Offset

CP Cyclic Prefix

C-RAN Cloud/Centralized - Radio Access Network

CRC Cyclic Redundancy Check

CRS Cell Specific Reference Signal

DAC Digital to Analog Converter

DCI Downlink Control Information

DSP Digital Signal Processor

eNodeB Evolved Node B

EPC Evolved Packet Core

FDD Frequency Division Duplex

FFT Fast Fourier Transform

FPGA Field Programmable Gate Array

GPP General Purpose Processor

GPS Global Positioning System

GSM Global System for Mobile Communications

ISI Inter-Symbol Interference

ISM Industrial Scientifical and Medical

IF Intermediate Frequency

IFFT Inverse Fast Fourier Transform

IoT Internet of Things

LNA Low Noise Amplifier

LPWAN Low Power Wide Area Networks

LSb Least Significant bit

LTE Long Term Evolution

MAC Medium Access Control

MIB Master Information Block

MIMO Multiple Input Multiple Output

MSb Most Significant bit

NPUSCH Narrowband Physical Uplink Shared Channel

NCCE Narrowband Control Channel Element

NRS Narrowband Reference Signal

NPDSCH Narrowband Physical Downlink Shared Channel

NPBCH Narrowband Physical Broadcast Channel

NPDCCH Narrowband Physical Downlink Control Channel

NB-IoT Narrowband Internet of Things

NI National Instruments

NPSS Narrowband Primary Synchronization Signal

NPRACH Narrowband Physical Random Access Channel

NSSS Narrowband Secondary Synchronization Signal

OFDM Orthogonal Frequency Division Multiplexing

PA Power Amplifier

PRB Physical Resource Block

PBCH Physical Broadcast Channel

PCCC Parallel Concatenated Convolutional Code

PDSCH Physical Downlink Shared Channel

PMCH Physical Multicast Channel

QPSK Quadrature Phase Shift Keying

RAN Radio Access Network

RRC Radio Resource Control

RNTI Radio Network Temporary Identifier

RF Radio Frequency

RV IDX Revolution Number

SAE System Architecture Evolution

SNR Signal-to-Noise Ratio

SDR Software Defined Radio

SIB System Information Block

TrCH Transport Channel

TTI Time Transmission Interval

TBS Transport Block Size

USRP Universal Serial Radio Peripheral

UHD USRP Hardware Driver

UE User Equipment

UMTS Universal Mobile Telecommunications System

Z-F Zero Forcing

Chapter 1

Introduction

This chapter, in order to contextualize the motivation for this dissertation, introduces the Narrowband Internet of Things (NB-IoT) standard concept. The proposed goals are presented followed by the dissertation's structure and contributions.

1.1 Framework

A road towards a "smarter" and more connected future is being paved with more and more daily objects capable of communicating within a network, providing new possibilities for interaction with day to day devices.

Internet of Things (IoT) is a cooperating and inter-combined network of devices such as vehicles, home appliances, sensors, etc. The network is ideally connected through wireless technologies. The IoT concept is becoming generalized for the end users, making common the use of remote monitoring, smart metering or even pet tracking on a daily basis. Connections with not only people but objects, vehicles and sensors in general through the Internet is now regular for millions of users thus promoting a continuous growth on IoT networks' densities.

In order to enable large scale networks for IoT, a new type of wireless protocols emerged named Low Power Wide Area Networks (LPWAN).

1.1.1 LPWAN

LPWAN technologies were designed to provide large area coverage, linking devices at low data rates, low bandwidth and with low power consumption.
Popular LPWANs such as LoRa [Sem17] and Sigfox [Sig17] operate in the Industrial Scientifical and Medical (ISM) bands with limited performance due to imposed constraints such as limited transmission power and reduced duty-cycle [Lab17]. Nevertheless, these services continue to seem appealing for project development due to their low price equipment. Open source communities such as The Things Network [Net17] continue to grow and invest in LoRa while designing their smart connected world.

In response to the growing LPWAN market, since June 2016 the initiative responsible for the Long Term Evolution (LTE) standard 3rd Generation Partnership Project (3GPP) released a new LPWAN standard able to operate on licensed bands, NB-IoT.

NB-IoT is planned to enter the same business market as LoRa and SigFox, as an LPWAN with low-cost devices, with the advantage of operating on a broader range of available bands without the ISM restrictions.

The architecture design for NB-IoT Radio Access Network (RAN) is similar to other cellular systems as shown in figure 1.1: An Evolved Node B (eNodeB), coordinating the cell to which the User Equipment (UE)s (devices) connect and inter change downlink (eNodeB to UE) and uplink (UE to eNodeB) messages. The collected data is sent through the Evolved Packet Core (EPC) to the Internet, as in LTE.

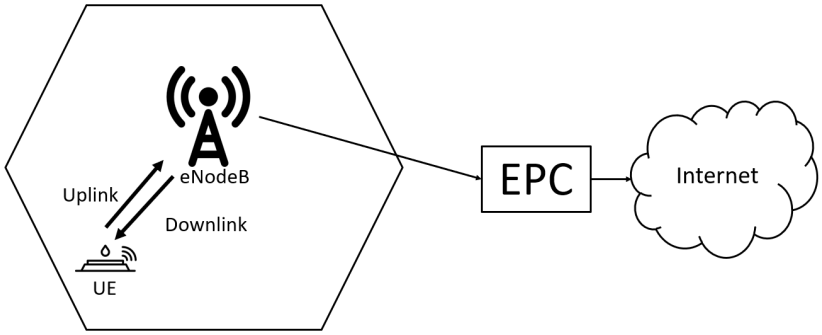


Figure 1.1: Example of NB-IoT RAN and message transmission.

Parameter	**LoRa**	**SigFox**	**NB-IoT**
Modulation	Chirp Spread Spectrum (CSS)	GFSK / BPSK	BPSK / QPSK
Operating Band	ISM	ISM	E-UTRA
Link Budget	150 - 157 dB	146 - 162 dB	$\sim$ 164 dB
Peak Data Rate Downlink	50kbps	600 bps	250 kbps
Peak Data Rate Uplink	50 kbps	100 bps	single tone-20 kbps multitone-250 kbps
Open Standard	Yes	No	Yes

Table 1.1: Comparison between LoRa, SigFox and NB-IoT Standards [Tec16].

Table 1.1 presents some main points for comparison of the three LPWANs. Regarding performance they are, overall, similar even though LoRa and NB-IoT provide higher data rates than SigFox. The main difference lays on the open standard feature.

The protocol stack for LoRa is open, i.e., it can be used and adapted for the user or developer's will and needs. The physical layer, however, is patented by Semtech. The deployment for a LoRa network only depends on the LoRa gateway's location, which is purchased and installed by the user and only depends on an Internet connection.

SigFox is a network service provider, the end-devices are purchased and to be used on a SigFox service there's an annual fee and can only be operated in SigFox service covered locations. The data collected is then delivered to the user over the SigFox service.

NB-IoT stands out on the scope of these three LPWAN since it's a completely open standard which developers can integrate on existing cellular networks.

For operators, NB-IoT is presented as a solution to enter the LPWAN market, providing new services for IoT deployment.

In February 2017 Vodafone started deploying NB-IoT commercial solutions in Spain [Vod17]. This deployment has been alongside other operators who believe that a 3GPP based LPWAN solution operating on the licensed band is a form of boosting the Cloud IoT concept for the connected world.

In [Nok16] the market placement for NB-IoT is presented as an enabler combining the key requirements for both LPWAN and Cloud IoT deployments for the expectancies in the growing market. In figure 1.2 Nokia presents the use cases for cellular IoT.

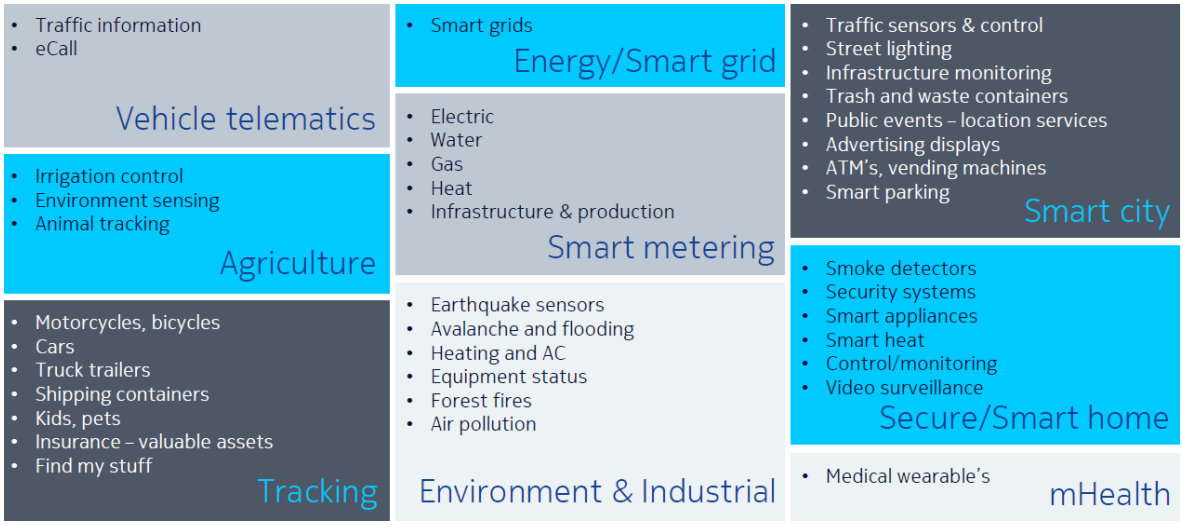


Figure 1.2: Cellular IoT use cases [Nok16]

1.2 Motivation

The standardization for NB-IoT is recent, having only been finalized in June 2016. For developing applications and hardware related to NB-IoT there are little and limited implementations for reference and verification testing.

The demand for NB-IoT based solutions such as flexible transceivers and protocol stacks calls for rapid development of both software and hardware solutions, eventually based on Software Defined Radio (SDR) developments .

To improve and facilitate development, flexible simulation and laboratory testbeds provide tools and references for proper testing, thus reducing time and hardship for first approaches.

A Matlab based simulator can be useful for hardware and application development on NB-IoT. Matlab presents a clear developing and testing environment and is fairly easily integrated with SDR development platforms such as the USRP [Res17b] .

Open source code implementations of NB-IoT Physical Layer, capable of performing the functions needed for signal generation and recovery, can be used for referencing, comparison and testing, reducing time and effort on future developments.

A co-simulation environment, with hardware in the loop, enables signal capture and its analysis can be done without real time constraints, which grants a significant ease while

approaching developing steps. The integration with hardware for transmitting using the air interface is also a tool for laboratory validation. This is an important step for future research regarding not only hardware development but also integration with existing systems and testbeds for new RAN concepts such as Cloud/Centralized - Radio Access Network (C-RAN).

1.3 Objectives

The project for this dissertation was designed in response to the identified needs and believing that NB-IoT will occupy a significant place on the LPWAN market. The proposed project aims to provide a toolbox and testbed for a first approach in NB-IoT hardware development.

In this dissertation is proposed a MATLAB testbed for NB-IoT's downlink physical layer in both simulation and co-simulation environments.

The final project is desired to be able to make end-to-end communications with NB-IoT's specifications. The design should implement a simulation environment to predict real life situations using different channel models to predict hazards and correct them.

The system should also integrate with a radio front-end in order build a co-simulation environment with hardware in the loop able to transmit and capture samples for laboratory validation.

1.4 Structure and Organization

The dissertation is distributed in 7 chapters.

- **Chapter 1 - Introduction** Presents the NB-IoT framework in the IoT market for other LPWAN. The objectives and motivation for this dissertation is presented. The contributions that this dissertation lead to are also mentioned
- **Chapter 2 - NB-IoT Physical Layer and General Concepts** Introduces the concepts needed for understanding the work developed. Overview of the 3GPP's standard in Release 13 for NB-IoT Physical Layer in downlink.
- **Chapter 3 - Matlab Modelling of the Downlink Transmitter** Describes the implementation made for generating NB-IoT downlink baseband samples.
- **Chapter 4 - Matlab Modelling of the Downlink Receiver** Describes the implementation made for recovering data out of the received NB-IoT downlink baseband samples.
- **Chapter 5 - RF Front End Integration and Co-simulation** Describes the implementation for the co-simulation environment using the USRP and the developed behavioral simulation for NB-IoT downlink receiver and transmitter.
- **Chapter 6 - Results** Presents the results for the tests the system was subjected to regarding different channel models. The results for the co-simulation environment are also presented.
- **Chapter 7 - Conclusions** Summarizes the developed work along with some considerations and future work suggestions.

- **Appendix A - User Guide** Describes how to run both the simulation an co-simulation environments.
- **Appendix B - Sequence Example** Presents an example of the generated signal's samples.

The final appendixes demonstrate how to operate the system and a sample outputs for every procedure regarding signal generation and recovery.

1.5 Original Contributions

This dissertation contributes with an open source, original and functional behavioral model for NB-IoT downlink in standalone mode for one antenna port.

A publication was submitted and is pending approval.

M. Brandão, **J. Nunes**, L. Almeida, J. Vieira and A. Oliveira "*Design of a NB-IoT Physical Layer Co-simulator*", submitted to REC XIV 2018 - Jornadas sobre Sistemas Reconfiguráveis - Monte da Caparica FCT-UNL.

Chapter 2

NB-IoT Physical Layer and General Concepts

This chapter makes an introduction to the concepts approached in this dissertation. An overview of the standard is presented explaining the need and purpose for each procedure in the Physical Layer.

2.1 Introduction

NB-IoT is a 3GPP radio access standard for IoT connectivity. Its first version was released in June of 2016 as part of 3GPP's Release 13. This technology is deployable in three different modes which are depicted in figure 2.1:

1. Standalone
   A Global System for Mobile Communications (GSM) carrier is refarmed for NB-IoT's purpose.

2. Guard Band
   The guard band of an LTE carrier is filled with an NB-IoT carrier.

3. In-band
   The NB-IoT carrier is inside an LTE carrier.

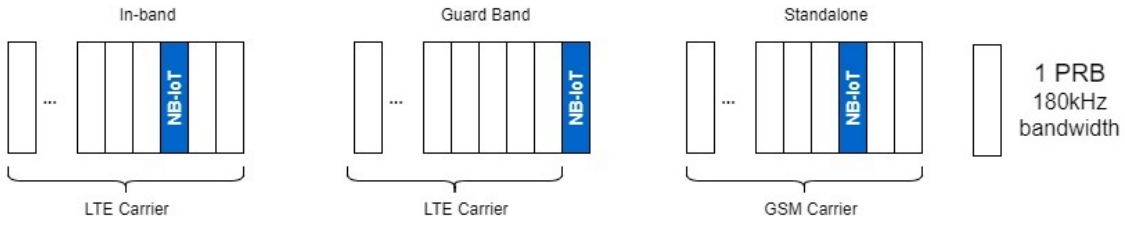


Figure 2.1: NB-IoT deployment modes.

2.2 NB-IoT Downlink Physical Layer

The Physical Layer is responsible for implementing logical procedures for transmission of data incoming from higher layers.

It interconnects with the Medium Access Control (MAC) layer receiving Transport Blocks with the data needed for transmission over the air interface as shown in figure 2.2.

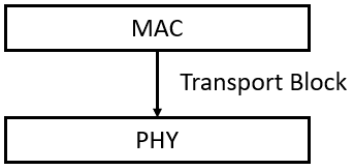


Figure 2.2: Interconnection between the MAC and PHY layer.

2.2.1 General Procedures

The general procedures describe the main blocks to which the upcoming Transport Blocks go through. Figure 2.3 depicts the procedures defined by 3GPP to generate baseband samples.

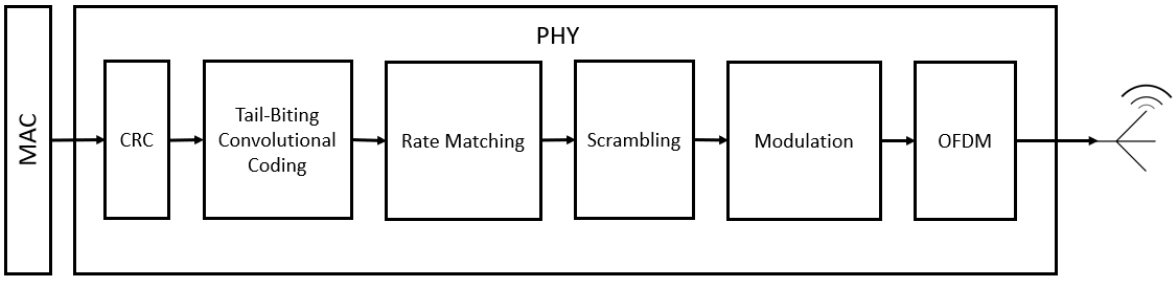


Figure 2.3: General Procedures of the Physical Layer.

2.2.1.1 CRC

To support forward error detection, the Transport Block is first appended with a Cyclic Redundancy Check (CRC) sequence in all bits using a generator polynomial of length $L \in \{8, 16, 24\}$ which are defined in table 2.1. CRC sequences provide robustness to the transmitted data enabling error detection.

Name	Generator Polynomial
24a	$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$
24b	$D^{24} + D^{23} + D^6 + D^5 + D + 1$
16	$D^{16} + D^{12} + D^5 + 1$
8	$D^8 + D^7 + D^4 + D^3 + 1$

Table 2.1: CRC generator polynomials defined in [3GP17a].

2.2.1.2 Tail-biting Convolution Coding

Encoding is applied to increase redundancy on the transmitted message and increasing the odds for correctly receive the transmitted data. The input data is encoded and therefore noted as a codeword.

As shown in figure 2.4, the "Tail Biting Convolutional Coding" code is defined with constraint length 7 and code rate 1/3.

Unlike other convolutional codes, tail biting coding does not use zero-trailing, i.e., it does not insert an m-length sequence of zeros in order to take the encoder back to all zeros state. Zero tailing is used because most convolutional encoders start and end the coding of input sequences in zero state. This process leads to code rate loss since the output sequence will be m bits larger than the input.

Tail biting convolutional codes start and end with the memory elements initialized by the first m bits of the input sequence, thus not needing the zero trailing.

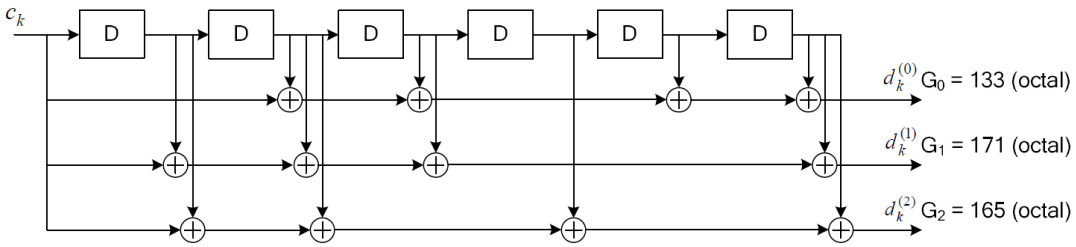


Figure 2.4: Tail biting convolutional encoder extracted from [3GP17a].

The three outputs are here forth noted as: $d^{(0)}$ - systematic bits, $d^{(1)}$ - parity1 bits and $d^{(2)}$ - parity2 bits.

2.2.1.3 Rate Matching

The Rate Matching procedure asserts the codeword to fit the payload size of the message.

For the rate matching procedure, sub-block interleaving is applied for each of three bitstream inputs. In each block the bits are then arranged into a matrix with C=32 columns and R rows. R is determined as the minimum integer to satisfy the condition $C \cdot R \geq D$, D being the number of input bits. If $D \mod C \neq 0$ dummy bits set as NULL shall be inserted. The obtained matrix is noted as $K\pi$.

For the interleaving process, the matrix columns are then permuted as defined in table 5.1.4-1 of [3GP17a].

The interleaving process provides robustness for forward error corrections. Errors usually occur in bursts which effect is reduced with interleaving since the codeword is permuted before transmission and then permuted back when receiving.

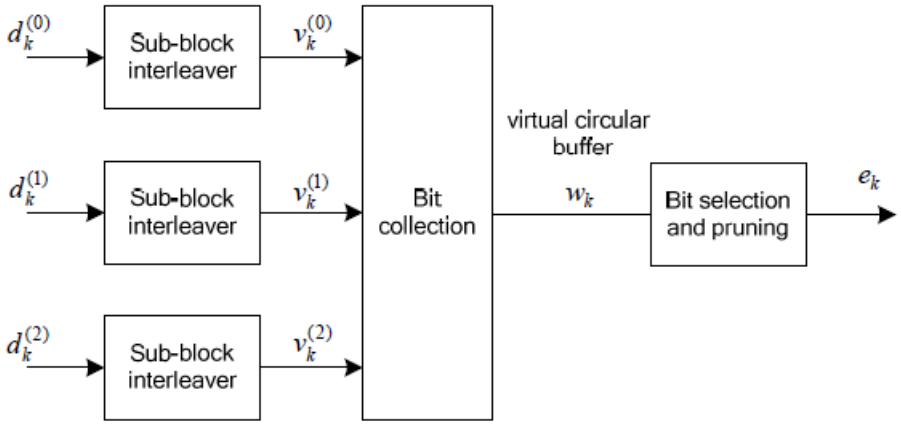


Figure 2.5: Rate Matching block diagram extracted from [3GP17a].

As shown in figure 2.5 the three matrices must now be collected and so bit collection is defined. A virtual circular buffer of size equal to $3 \cdot K\pi$ is generated to concatenate the three matrices.

The virtual buffer is arranged with the first $K\pi$ bits being the systematic bits and in the following $2 \cdot K\pi$ bits all even bits belong to the parity1 sub-block and the odd bits to the parity2 sub-block. The result is shown in

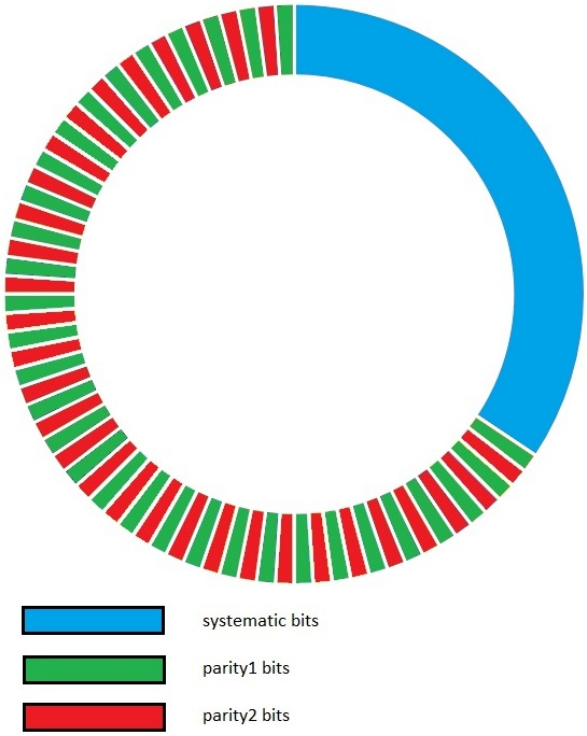


Figure 2.6: Circular buffer of the collected interleaved matrixes of systematic, parity1 and parity2 bits.

The bits now need to be selected, to provide different code block subsets for different packet transmissions, hence bit selection is defined. This process, puncturing, provides an output sequence with different subsets for different transmissions of a packet.

2.2.1.4 Scrambling

Scrambling encodes the code words bitwise for protection to burst errors and inter-cell interference. It's a form of encryption which makes the message unreadable for any device unknowing the scrambling sequence. The procedure also ensures that long sequences of zeros or ones aren't transmitted.

The scrambling process is based on a bitwise *XOR* between the input sequence and a length-31 Gold Sequence, to provide low cross-relation within sets and allow for multiple users while broadcasting.

A Gold sequence, or a gold code, is a spreading sequence for spread spectrum systems [Wav17b].

The Gold Sequence is generated by the *XOR* operation between two m-sequences as shown in figure 2.7.

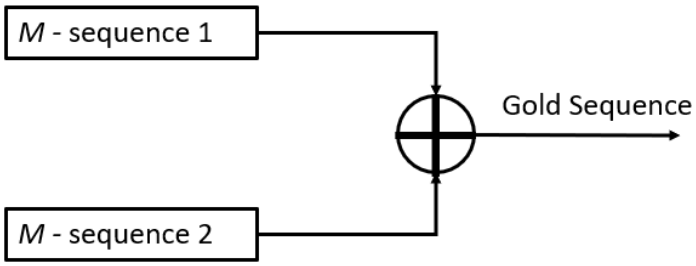


Figure 2.7: Example of a Gold Sequence generator block diagram.

Gold sequences present worst auto-correlation properties but better cross-correlation properties when compared to m-sequences.

In [3GP16] the gold sequence output is denoted as c and the m-sequences are defined as x_1 and x_2. The equations for generating the gold sequence are shown in (2.1), (2.2) and (2.3).

$$c(n) = (x_1(n + N_c) + x_2(n + N_c)) \mod 2 \tag{2.1}$$

$$x_1(n + 31) = (x_1(n + 3) + x_1(n)) \mod 2 \tag{2.2}$$

$$x_2(n + 31) = (x_2(n + 3) + x_2(n + 2) + x_2(n + 1) + x_2(n)) \mod 2 \tag{2.3}$$

Where N_C=1600 and the m-sequence x_1 is always initialized with $x_1(0) = 1$ and $x_1(1 : 30) = 0$. The x_2 sequence is initialized depending on the application for the gold sequence.

2.2.1.5 Modulation

The modulation procedure converts the scrambled bits into complex-valued symbols.

For NB-IoT all downlink messages are modulated in Quadrature Phase Shift Keying (QPSK) symbols. For every two bits one symbol is generated.

Input	Output
00	$1/\sqrt{2}$ + j$1/\sqrt{2}$
01	$1/\sqrt{2}$ - j$1/\sqrt{2}$
10	-$1/\sqrt{2}$ + j$1/\sqrt{2}$
11	-$1/\sqrt{2}$ - j$1/\sqrt{2}$

Table 2.2: QPSK modulation table extracted from [3GP16].

2.2.2 OFDM Signal Generation

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation technique used for wireless communications. In this technique, instead of using a single carrier, the data is multiplexed into several subcarriers within the channel's bandwidth.

The subcarriers are orthogonal to each other since they don't overlap, thus not interfering with each other. OFDM's main advantage is that the same bit rate as single carrier modulations can be achieved while providing more robustness to the signal's deterioration due to the channel effect.

The input data is modulated with simpler forms such as QPSK, for NB-IoT. The QPSK symbols are then multiplexed in frequency domain, assigned a subcarrier.

Figure 2.8 presents an example of OFDM modulation.

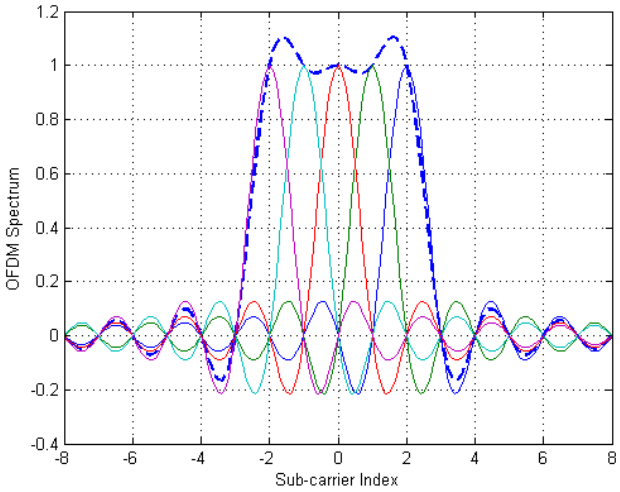


Figure 2.8: Example of OFDM modulation [Nut17].

In NB-IoT for downlink OFDM is used and the subcarriers are separated by $\Delta f = 15kHz$.

2.2.3 Frame Structure

In NB-IoT frames are built similar to LTE. The difference is based on the slots and their. Slots are made from the result of the resource grid, which is now limited to one resource block, as shown in figure 2.9.

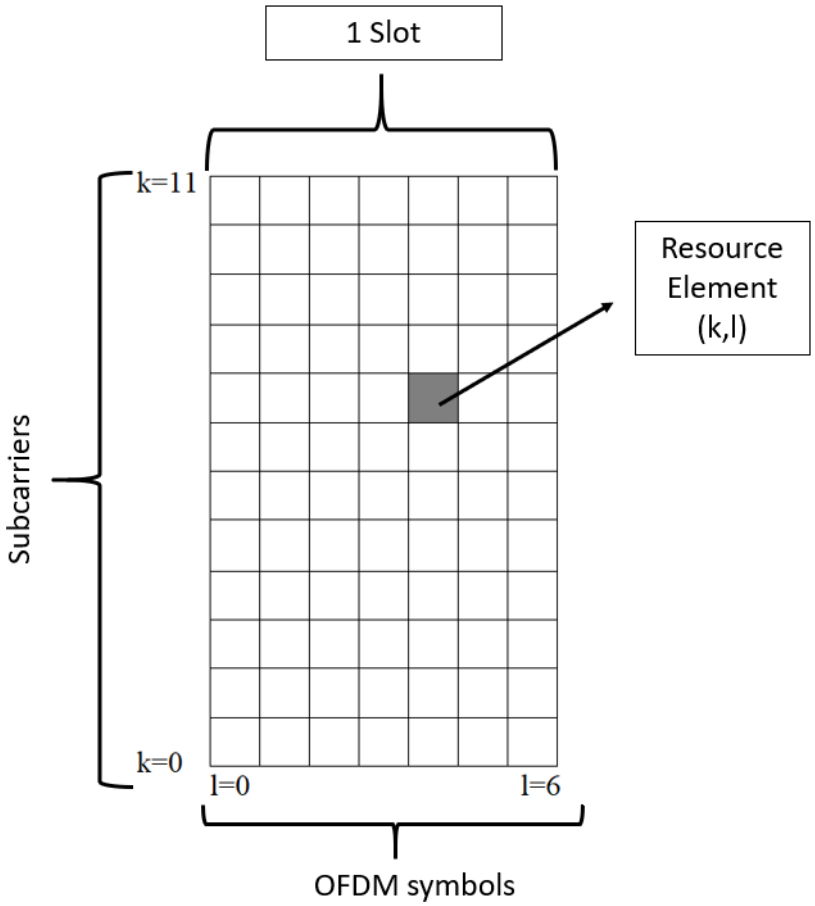


Figure 2.9: NB-IoT resource grid, equivalent to one PRB.

The limitation regarding the resource grid size defines some radio features, namely the bandwidth. For downlink every transmission occupies a full Physical Resource Block (PRB) with $\Delta f = 15kHz$ between subcarriers, hence the bandwidth is fixed to 180kHz.

NB-IoT uses only Frequency Division Duplex (FDD) and as defined in [3GP16] only frame structure type 1 is applied.

One PRB makes a slot, and two consecutive slots form a subframe. Every 10 subframes form a radio frame as shown in figure 2.10.

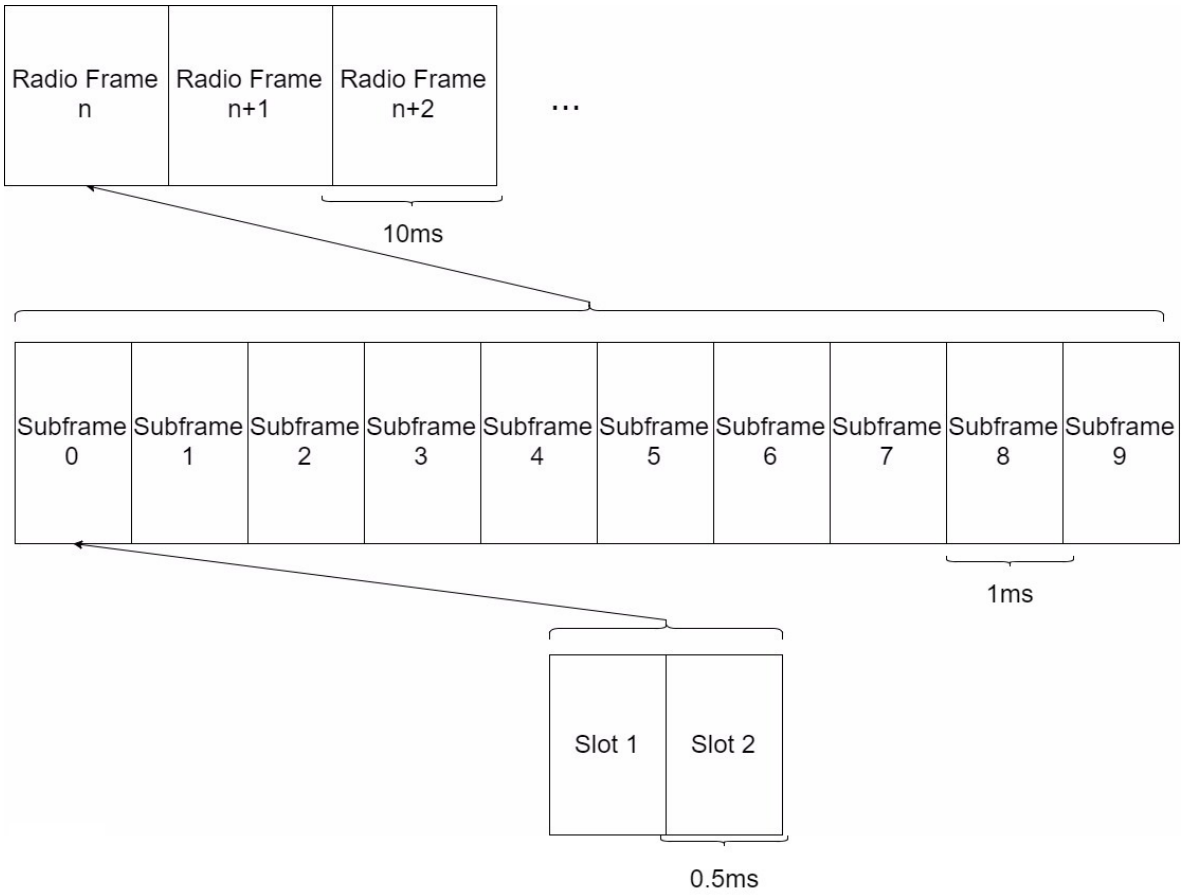


Figure 2.10: Frame Structure Type 1 block diagram. Example of the construction of subframes and radio frames.

2.2.4 Physical Channels and Signals

Similar to LTE, NB-IoT's Physical layer integrates physical channels and signals. Physical channels carry the data provided from the MAC layer's transport channels through transport blocks. Physical signals are only used by the Physical Layer to assist the transmission and reception of the physical channels.

For NB-IoT in downlink there are three physical channels:

- Narrowband Physical Broadcast Channel (NPBCH)
- Narrowband Physical Downlink Control Channel (NPDCCH)
- Narrowband Physical Downlink Shared Channel (NPDSCH)

There also three physical signals:

- Narrowband Reference Signal (NRS)
- Narrowband Primary Synchronization Signal (NPSS)

- Narrowband Secondary Synchronization Signal (NSSS)

Transport Blocks are coded and later mapped to physical channels. The mapping of physical channels in NB-IoT is shown in figure 2.11.

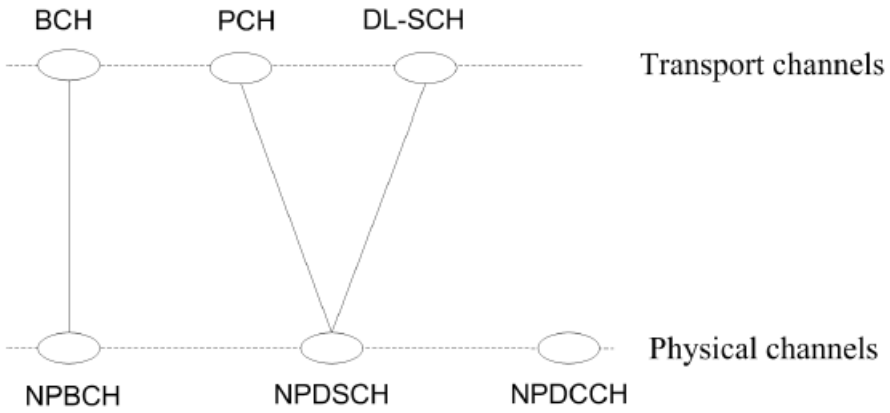


Figure 2.11: Mapping of Transport Channels to Physical Channels in NB-IoT [SR16].

2.2.4.1 NPBCH

NPBCH is similar to LTE's Physical Broadcast Channel (PBCH). It carries the Master Information Block (MIB) which is vital information for cell access and system information. The values carried are: [3GP17c]

- 4 bits indicating the Most Significant bit (MSb) of the system frame number
- 2 bits indicating the two Least Significant bit (LSb) of the hyper frame number
- 4 bits for the SIB1-NB scheduling and size
- 5 bits indicating the system information value tag
- 1 bit indicating whether access class barring is applied
- 7 bits indicating the operation mode with the mode specific values
- 11 spare bits for future extensions

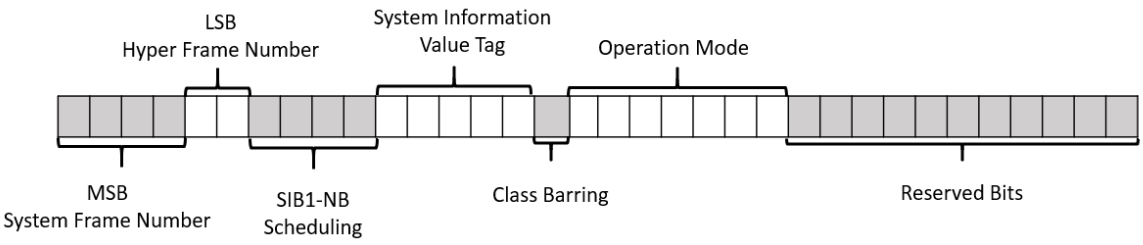


Figure 2.12: Schematics of the MIB bitstream

2.2.4.2 NPDCCH

The NPDCCH is the control channel for NB-IoT transmissions, responsible for carrying Downlink Control Information (DCI). In NB-IoT there are three DCI formats: [3GP17a]

- Format N0

  Used for scheduling the uplink device (NPUSCH)

- Format N1

  Used for scheduling the NPDSCH and the uplink device (NPRACH)

- Format N2

  Used for paging and direct indication

DCI Format N0

Format N0 is assigned for each uplink device with the purpose of controlling and scheduling the Narrowband Physical Uplink Shared Channel (NPUSCH). The N0 message comprehends 23 bits with the following content [3GP17b]:

■ 1 bit - Format Flag indication ('0' indicates N0 , '1' indicates N1)

■ 6 bits - Subcarrier indication , I_{sc} Indicates the uplink device which set of subcarriers to use. The set of possible values is shown in table 2.3.

I_{sc}	n_{sc}
0-11	I_{sc}
12-15	$3(I_{sc}\text{-}12)+\{0,1,2\}$
16-17	$6(I_{sc}\text{-}16)+\{0,1,2,3,4,5\}$
18	$\{0,1,2,3,4,5,6,7,8,9,10,11\}$
19-63	Reserved

Table 2.3: Set of allocated subcarriers for NPUSCH transmission - Table 16.5.1.1-1 extracted from [3GP17b] .

■ 3 bits - Resource assignment ,I_{RU} Assigns the number of resource units to be used for the NPUSCH. The possible values are shown in table 2.4

I_{RU}	N_{RU}
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

Table 2.4: Number of Resources Units to be used by NPUSCH transmission - Table 16.5.1.1-2 extracted from [3GP17b].

■ 2 bits - Scheduling delay , I_{Delay}. Subframe indicator for NPUSCH transmission. The possible values are shown in table 2.5.

I_{Delay}	k_0
0	8
1	16
2	32
3	64

Table 2.5: The starting slot for the uplink transmission is given by n+k_0 where n is the current NPDCCH subframe - Table 16.5.1-1 of [3GP17b].

■ 4 bits - Modulation and coding scheme I_{MCS} - Indicates the modulation to use in uplink transmission, Q_m, and the transport block size, I_{TBS}. The possible values are showin in table 2.6

I_{MCS}	Q_m	I_{TBS}
0	1	0
1	1	2
2	2	1
3	2	3
4	2	4
5	2	5
6	2	6
7	2	7
8	2	8
9	2	9
10	2	10

Table 2.6: Modulation and Coding Scheme for Uplink Transmission - Table 16.5.1.2-1 extracted from [3GP17b].

■ 1 bit - Redundancy version - Flag indicator

■ 3 bits - Repetition number , I_{Rep} - Number of NPUSCH repetitions. The possible values are shown in table 2.7

I_{Rep}	N_{Rep}
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Table 2.7: Number of repetitions for NPUSCH transmission - Table 16.5.1.1-3 of [3GP17b].

■ 1 bit - New data indicator - Flag indicator
■ 2 bits - DCI subframe repetition number

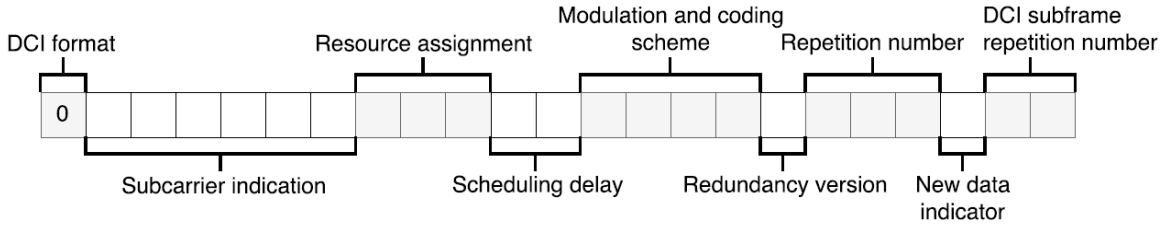


Figure 2.13: Example of DCI format N0 message bitstream [TK17]

DCI Format N1

For each NPDSCH codeword a format N1 message is assigned containing its scheduling information. Format N1 also schedules the uplink device for Narrowband Physical Random Access Channel (NPRACH) transmission. The information is passed down in 23 bits as follows [3GP17b]:

■ 1 bit - Format Flag indication ('0' indicates N0 , '1' indicates N1)
■ 1 bit - NPDCCH order indicator

- If NPDCCH order is set to '1' and CRC is scrambled with C-Radio Network Temporary Identifier (RNTI) then the N1 message is set to schedule the NPRACH with the following fields:

  - 2 bits - starting number of NPRACH repetitions, shown in table 2.8

I_{Rep}	N_{Rep}
0	R1
1	R2
2	R3
3	reserved

Table 2.8: Number of NPRACH repetitions, where R1,R2 and R3 are given by higher layer parameters - Table 16.3.2-1 extracted from [3GP17b] .

  - 6 bits - NPRACH subcarrier indication, I_{sc}.(Table 2.3)

  - The remaining 13 bits are set to 1

- If it's set to '0' then the N1 message is set for NPDSCH scheduling

  - 3 bits - Scheduling delay I_{Delay}. The values are shown in 2.9

I_{Delay}	k_0	
	$R_{max} < 128$	$R_{max} \geq 128$
0	0	0
1	4	16
2	8	32
3	12	64
4	16	128
5	32	256
6	64	512
7	128	1024

Table 2.9: Scheduling delay indicator for a given delay before NPDSCH transmission - Table 16.4.1.1-1 of [3GP17b]

- 3 bits - Resource Assignment, I_{SF} - Determines the number of available subframes for NPDSCH transmission. Table 2.10 shows the number of available subframes for one transmission.

I_{SF}	N_{SF}
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

Table 2.10: Number of subframes (N_{SF}) for NPDSCH - Table 16.4.1.3-1 of [3GP17b].

- 4 bits - Modulation and Coding Scheme, I_{MCS}- Determines the NPDSCH Transport Block Size (TBS). All downlink transmissions are QPSK modulated so the modulation order $Q_m = 2$ is assumed. To determine the TBS $I_{TBS} = I_{MCS}$ and is given as input to table 2.11. If the *SystemInformationBlockType1-NB* is being carried then the TBS is determined by table 2.12 with I_{TBS} set by higher layer parameters carried in the MIB.

I_{TBS}	I_{SF}							
	0	1	2	3	4	5	6	7
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	
6	88	176	256	392	504	600		
7	104	224	328	472	584	680		
8	120	256	392	536	680			
9	136	296	456	616				
10	144	328	504	680				
11	176	376	584					
12	208	440	680					

Table 2.11: NPDSCH Transport Block size - Table 16.4.1.5.1-1 extracted from [3GP17b].

I_{TBS}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TBS	208	208	208	328	328	328	440	440	440	680	680	680	reserved			

Table 2.12: TBS table for NPDSCH carrying SystemInformationBlockType1-NB - Table 16.4.1.5.2-1 extracted from [3GP17b].

- 4 bits - Repetition number. The possible values are shown in table 2.13

I_{Rep}	N_{Rep}
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	192
9	256
10	384
11	512
12	768
13	1024
14	1536
15	2048

Table 2.13: Number of repetitions (N_{Rep}) for NPDSCH - Table 16.4.1.3-2 of [3GP17b]

- 1 bit - New Data Indicator (reserved if CRC is scrambled with RA-RNTI)

- 4 bits - HARQ-ACK Resource (reserved if CRC is scrambled with RA-RNTI)

- 2 bits - DCI subframe repetition number

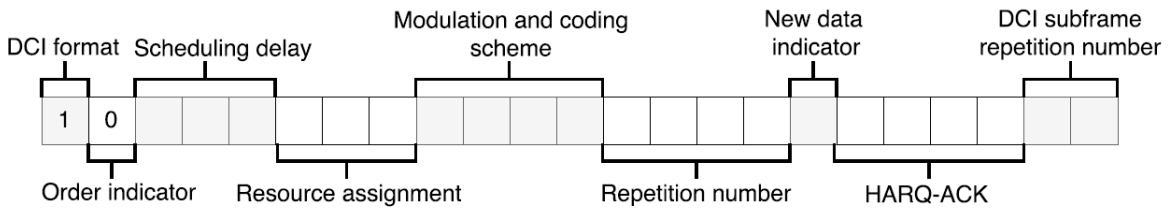


Figure 2.14: Example of DCI format N1 message bitstream for scheduling the NPDSCH [TK17]

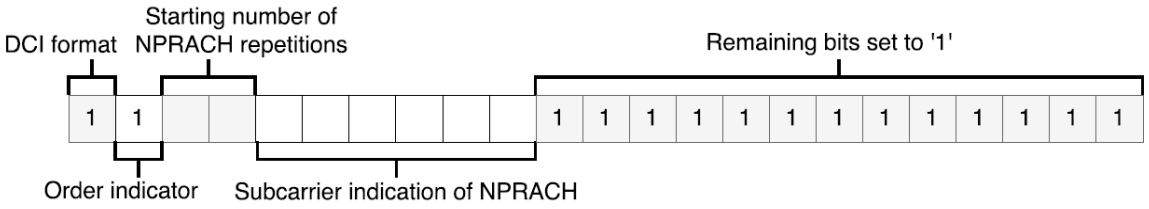


Figure 2.15: Example of DCI format N1 message bitstream for Random Access Procedures [TK17]

DCI Format N2

DCI format N2 is used for paging and direct indication, for informing the UE of parameter changes or to issue warning messages. Format N2 comprehends 15 bits, with the information as follows: [3GP17b]

■ 1 bit - Flag for paging/direct indication differentiation , with value 0 for direct indication and value 1 for paging

- '0' sets the N2 message for Direct Indication with the following paramenters

  - 8 bits - Direct Indication information provide direct indication of system information update and other fields

  - Reserved bits added to ensure the number of bits in the N2 message equals 15

- '1' sets the N2 message for Paging - 3 bits - Resource assignment, I_{SF}

  - 4 bits - Modulation and coding scheme, I_{MCS}

  - 4 bits - Repetition number

  - 3 bits - DCI subframe repetition number

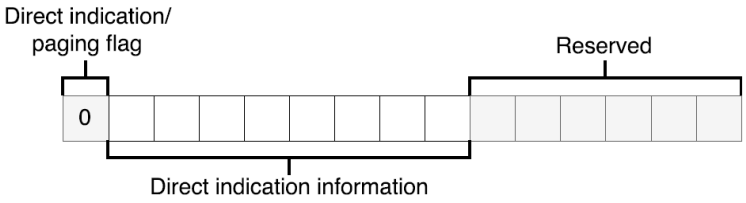


Figure 2.16: Example of DCI format N2 message bitstream for direct indication [TK17]

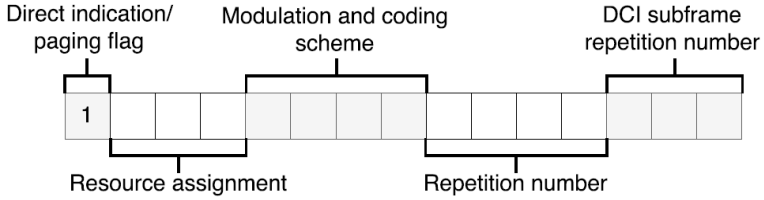


Figure 2.17: Example of DCI format N2 message bitstream for paging [TK17]

2.2.4.3 NPDSCH

The NPDSCH is the data channel for NB-IoT downlink. This channel is responsible for carrying user data and signaling information not sent through the NPBCH such as the System Information Block (SIB). Unlike LTE and other NPDSCH messages, when SIB1-NB is scheduled, its scheduling information is provided by the MIB.

System Information Block In NB-IoT there are 7 types of SIB, listed as follows [3GP17c]:

1. **SystemInformationBlock Type 1 - NB** - SIB1-NB - contains the scheduling information for the remaining System Information Blocks. It also contains information regarding the UE's permission to access the cell.
2. **SystemInformationBlock Type 2 - NB** - SIB2-NB - contains the information element regarding Radio Resource Control (RRC) which is common for all UEs in the cell.
3. **SystemInformationBlock Type 3 - NB** - SIB3-NB - contains the informationg element carrying detail for cell re-selection for intra-frequency and interfrequency cell re-selection.
4. **SystemInformationBlock Type 4 - NB** - SIB4-NB - contains information regarding neighbouring cell relevant only for intrafrequency cell re-selection.
5. **SystemInformationBlock Type 5 - NB** - SIB5-NB - contains information regarding NB-IoT neighbouring cell frequencies.
6. **SystemInformationBlock Type 14 - NB** - SIB14-NB - contains Access Barring parameters.
7. **SystemInformationBlock Type 16 - NB** - SIB16-NB - contains Global Positioning System (GPS) and time information which the UE may acquire.

2.2.4.4 NRS

The NRS is similar to LTE's Cell Specific Reference Signal (CRS). It's used for channel estimation and frame synchronization. Every physical channel is mapped around the NRS.

The complex values for the NRS are generated with equation (2.4).

$$r_{l,n_s}(m) = \frac{1}{\sqrt{2}}(1 - 2 \cdot c(2m)) + j\frac{1}{\sqrt{2}}(1 - 2 \cdot c(2m+1)), m = 0, 1 \tag{2.4}$$

$c(m)$ is a gold sequence and it's initialized with (2.5).

$$c_{init} = 2^{10} \cdot (7 \cdot (n_s + 1)) + l + 1) \cdot (2 \cdot N_{ID}^{cell} + 1) + 2 \cdot N_{ID}^{cell} + 1 \tag{2.5}$$

Where l is the OFDM symbol within the slot and n_s is the slot number within the radio frame.

2.2.4.5 NPSS and NSSS

Similar to LTE, NB-IoT uses two synchronization signals. They are used for frame timing offset compensation and to obtain the cell identity, N_{ID}^{cell}.

The generation of the complex values for the NPSS and NSSS is made by the output of a Zadoff-Chu sequence.

Zadoff-Chu Sequence Zadoff-Chu sequence, or also referred as Chu sequence or Frank-Zadoff-Chu sequence, is a complex-valued sequence. It is a Constant Amplitude Zero Autocorrelation (CAZAC) sequence used for wireless communications.

The Zadoff-Chu sequence has constant amplitude, zero circular autocorrelation, flat frequency domain response and has low cross relation between other Zadoff-Chu sequences. [Wor17]

The complex values for the Zadoff-Chu Sequence are calculated using equation (2.6).

$$e^{-j\frac{\pi un(n+1)}{N_c}} \tag{2.6}$$

An example of the output of a Zadoff-Chu sequence is shown in figure 2.18.

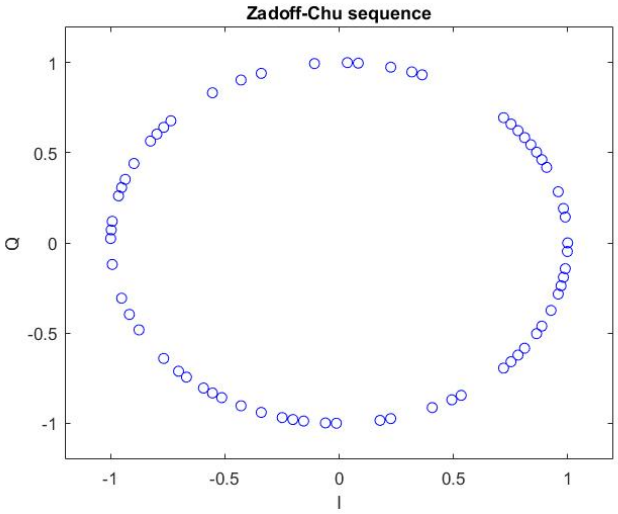


Figure 2.18: Example of Zadoff-Chu sequence for u=1, k=1,2...200 and Nc=131.

The presented physical layer procedures were thoroughly followed and implemented in a Matlab environment for the generation of baseband samples.

The same procedures were taken in account to perform the symmetric function for the recovery of transmitted data from the baseband samples. The implementations are presented in the following chapters.

Chapter 3

Matlab Modelling of the Downlink Transmitter

This chapter presents the Matlab modelling and implementation of the NB-IoT downlink transmitter. The implemented chain and procedures are demonstrated along with sample results.

3.1 Introduction

The transmitter side is responsible for the data coding and modulation steps for generating the baseband samples. These are implemented in *generateDLsignal.m*. The script both gathers and computes the data in order to build the baseband samples of a radio frame. The baseband samples generated are standard compliant for one antenna port in standalone mode. The transmitter process can be shown in figure 3.1.

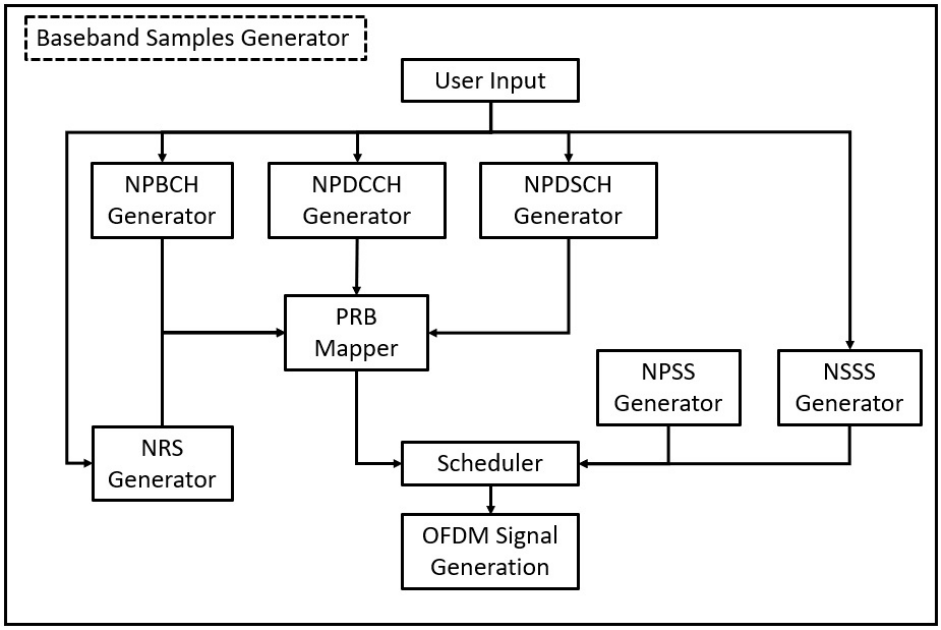


Figure 3.1: Implemented transmitter macro-block diagram.

3.2 General Procedures

The script is designed to operate with either pre-defined or user defined base station settings such as Cell ID, RNTI and Revolution Number (RV IDX) provided by the function $userInput()$.

3.2.1 Coding and Modulation

For NB-IoT channel generation coding and modulation procedures must be applied. The script codes and modulates the data for each channel individually and then maps and schedules it together to form a radio frame.

The coding procedures presented in section 2.1 were thoroughly followed and applied in various scripts as decribed below. The general blocks can be shown in figure 3.2.

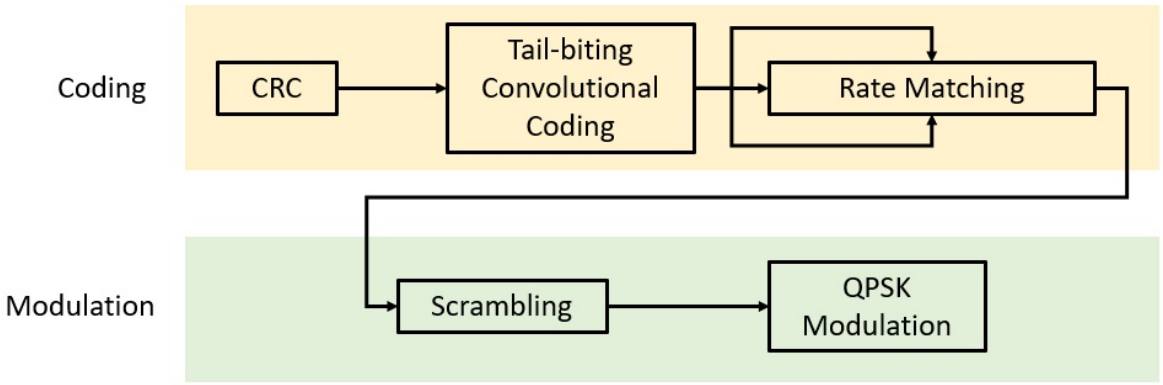


Figure 3.2: Coding and Modulation procedures.

3.2.1.1 CRC

The script *addcrc.m* calculates the CRC appendix, delivering a sequence with the added appendix on the output, by making multiple divisions with the input bits and the desired CRC mask polynomial. The remainder of the division is the CRC sequence to append.
In [3GP16] the generator polynomials are defined as in table 2.1. An example of the implemented code is presented in figure 3.3. The Matlab system call for the `addcrc` function is presented in table 3.1.

Function	`addcrc`
Input	`input - bitstream to be appended`
	`poly - generator polynomial`
Output	`addedbits - CRC appended output`
	`K - output length`

Table 3.1: addcrc function description.

To facilitate the process the scripts $crc24a.m$ and $crc16pbch.m$ were created to provide an easier and quicker system call which are presented in tables 3.2 and 3.3.

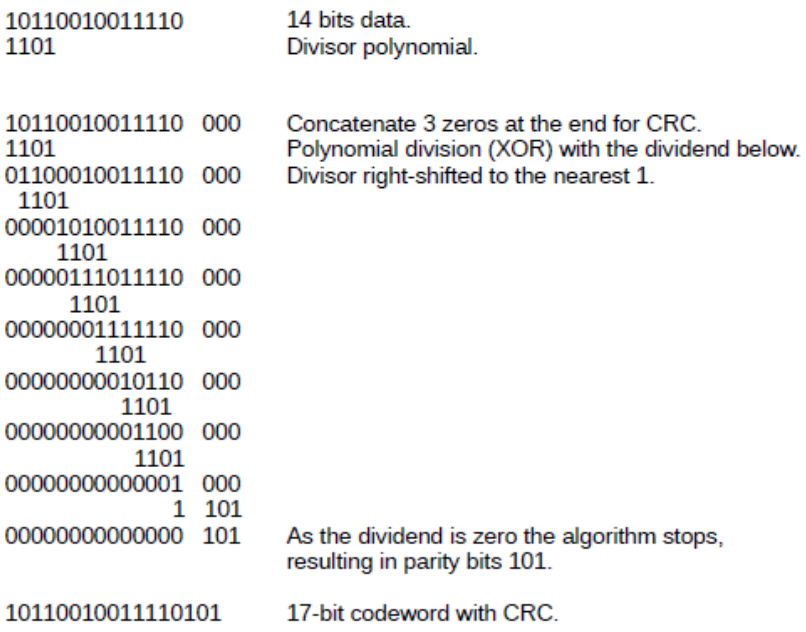


Figure 3.3: Example the CRC sequence calculation algorythim [TK17].

Function	`crc16pbch`
Input	`input - bitstream to be appended`
Output	`bits - CRC appended bits`
	`K - output length`

Table 3.2: crc16pbch function description. In this function the 16 bit polynomial for the BCH in one antenna port is already defined and and set as input to `addcrc.`

Function	`crc24a`
Input	`input - bitstream to be appended`
Output	`bits - CRC appended bits`
	`K - output length`

Table 3.3: crc24a function description. In this function the 24A polynomial is already defined and and set as input to `addcrc.`

3.2.1.2 Tail-biting convolutional coding

The tail-biting convolutional coding was implemented in script *tailbitingcc.m*. The encoding is made by initializing 6 memory states with the last 6 bits from the input bitstream which define the stable state of the encoder. The upcoming bits, from first to last, will then update the memory states and for each iteration a new output is calculated as shown in 3.4. For output $d^{(0)}$ equation (3.1) is applied, (3.2) for $d^{(1)}$ and (3.3) for $d^{(2)}$.

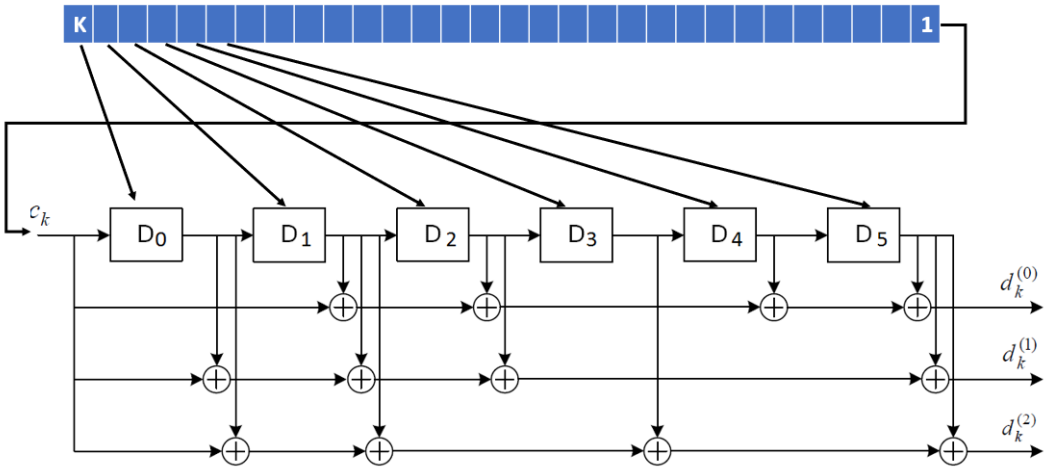


Figure 3.4: Tail-biting Convolutional Coding sequence.

$$d_0 = D_5 \oplus D_4 \oplus c_k \oplus D_1 \oplus D_2 \tag{3.1}$$

$$d_1 = D_5 \oplus D_2 \oplus c_k \oplus D_1 \oplus D_0 \tag{3.2}$$

$$d_2 = D_5 \oplus D_3 \oplus c_k \oplus D_1 \oplus D_0 \tag{3.3}$$

Figure 3.5 shows an example of the three encoded outputs. The output d_0 is the systematic bits, d_1 the parity 1 bits and d_2 the parity2 bits.

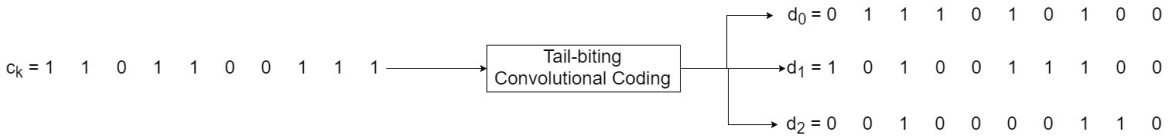


Figure 3.5: Example of an output of the Tail-biting Convolutional Coding.

The function receives as input the bitstream to code and the bitstream's length and delivers three bitstreams on the output. The developed Matlab function's system call is presented in table 3.4.

Function	`tailbitingcc`
Input	`input - bitstream to be appended`
	`K- input length bits`
Output	`d_0 - systematic bits`
	`d_1 - parity 1 bits`
	`d_2 - parity 2 bits`

Table 3.4: tailbitingcc function description.

3.2.1.3 Rate Matching

The rate matching procedure is implemented in script *rateMatching.m* and has three internal steps.
To perform the rate matching firstly the sub-block interleaving must be applied. This routine is made by the script *subblock_interleaver.m*. This function first assures the input bitstream sizes a matrix of 32 columns by R rows, filling the stream with $< NULL >$ if necessary. The procedure then changes for the different inputs. For the inputs with index 0 and 1 the permutation matrix applied is P.

P=[1,17,9,25,5,21,13,29,3,19,11,27,7,23,15,31,0,16,8,24,4,20,12,28,2,18,10,26,6,22,14,30]

For bitstreams with index 2 the interleaving is applied by sorting the bits according (3.4).

$$\pi(k) = \left(P\left(\left\lfloor \frac{k}{R^{TC}_{subblock}} \right\rfloor \right) + C^{TC}_{subblock} \times \left(k mod R^{TC}_{subblock} \right) + 1 \right) mod K_{\prod} \tag{3.4}$$

The call for the `subblock_interleaver` function is presented in table 3.5.

Function	`subblock_interleaver`
Input	`d - input coded bitstream`
	`superscript - index of the input bitstream`
Output	`v - interleaved outptut`

Table 3.5: subblock_interleaver function description.

For example, for the input in figure 3.6 , with superscript 1.

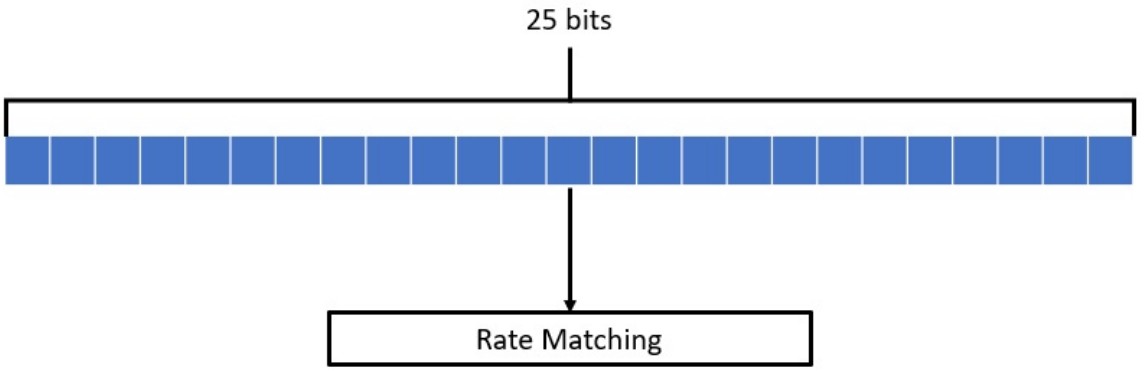


Figure 3.6: Input example for the Rate Matching procedure.

The function will first assert the size of the bitstream and fit it to a matrix of 32 columns filling it with $< NULL >$ as shown in 3.7:

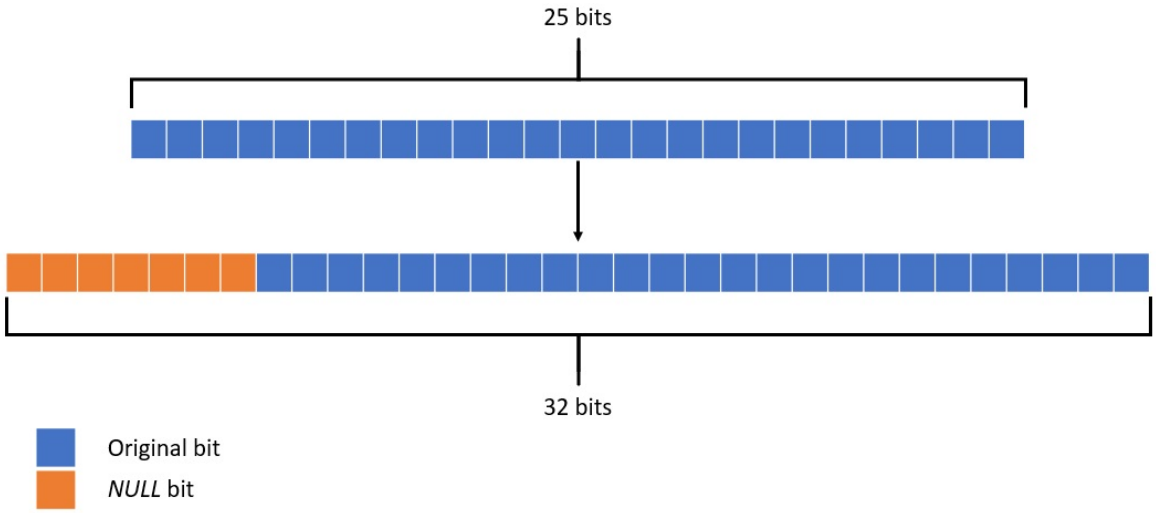


Figure 3.7: Row fitting in subblock interleaving procedure.

After fitting, the permutation is then applied resulting in the matrix shown in 3.8.

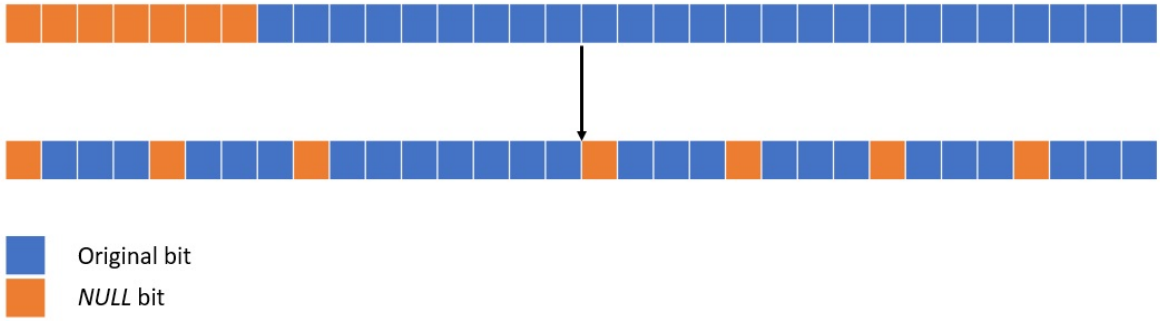


Figure 3.8: Matrix permutation based on superscript 1 of subblock interleaving procedure.

The circular buffer is then built using Matlab's reshape function, concatenating the three bitstreams (d_0, d_1, d_2) accordingly as shown in figure 2.6.

The last procedure for the rate matching routine is the puncturing, elaborated in *puncturer.m* [Mau17].

In the puncturing procedure the rate matching output is built by selecting the bits from the circular buffer ordered by a puncturing pattern defined in [3GP17a], filtering out the $< NULL >$ bits. The system call for the puncturer is presented in table 3.6.

Function	puncturer
Input	K - output length
	E - input length
	w - circular buffer
Output	punctured - punctured output

Table 3.6: puncturer function description.

Function	rateMatching
Input	d_0 - systematic bits
	d_1 - parity 1 bits
	d_2 - parity 2 bits
	G - output length
	rv_idx - revolution number
Output	e - rate matched output

Table 3.7: rateMatching function description.

3.2.1.4 Scrambling

The Scrambler applies an *XOR* to the input bits with the calculated gold sequence. The gold sequence is calculated in *goldseq.m*. The script receives as inputs the initialization polynomial, $x2\_init$, and the output length, Mpn. The procedure is implemented inline with the code.

Function	goldseq
Input	x2_init - initialization sequence
	Mpn - sequence length
Output	c - gold sequence output

Table 3.8: goldseq function description.

The scrambling procedure is examplified in figure 3.9:

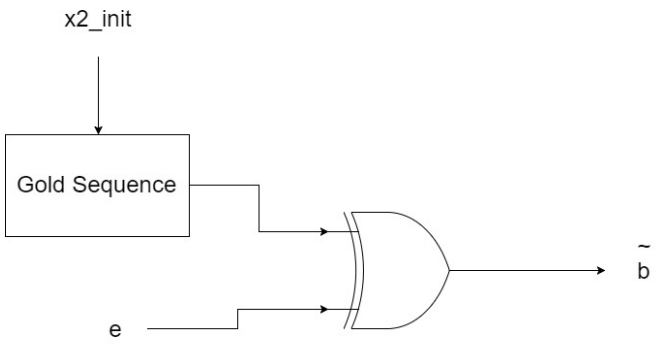


Figure 3.9: Scrambler block diagram.

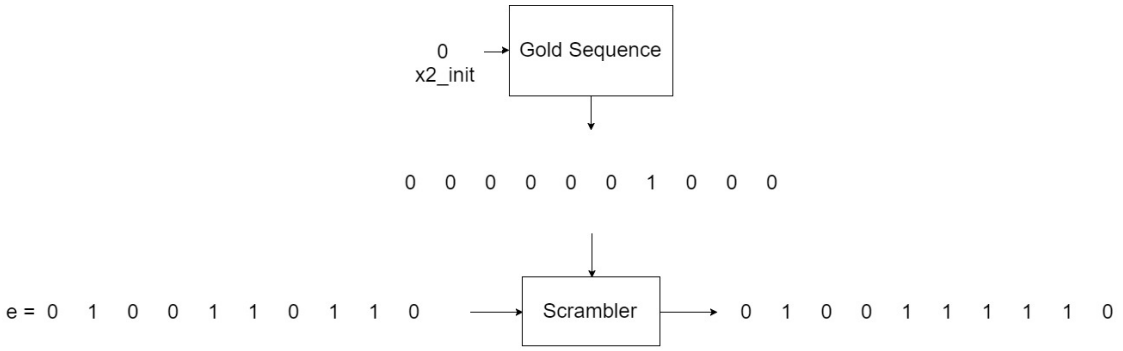


Figure 3.10: Scrambler example output.

3.2.1.5 Modulation

The modulation procedure is in charge of converting bits into complex numbers. The wanted modulation is exclusively QPSK so the code in script *mod_data.m* pairs every two bits and outputs the corresponding complex number according to 2.2.

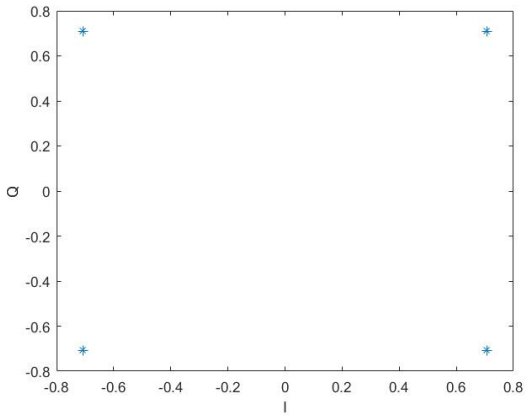


Figure 3.11: Example of QPSK constellation diagram.

Function	`mod_data`
Input	`data - input data to modulate`
	`Qm - modulation order`
Output	`data_symbol - modulated data`

Table 3.9: mod_data function description. data is the input bitstream and Qm the modulation scheme.

3.3 Physical Signals Generation

As stated in above sections, the modulated complex numbers are then grouped in slots, PRB, which then form sub-frames and radio frames. PRBs containing Physical Channels must include the NRS and to form a radio frame. Subframes containing the Synchronization Signals are created separately.

3.3.1 Narrowband Reference Signal

For every physical channel, when mapped into a resource block, the NRS must be applied and placed in pre-defined resource elements.

The script *referenceSignalFrame.m*, described in table 3.11, generates one resource block with the pre-allocated NRS resource elements. The NRS values are calculated in *referenceSignalGenerator.m*, described in table 3.10 as a result of a gold sequence.

The physical channels' complex values are then used to fill the remaining resource elements in the resource block.

The positions occupied by the NRS, for one antenna port, are shown in figure 3.12

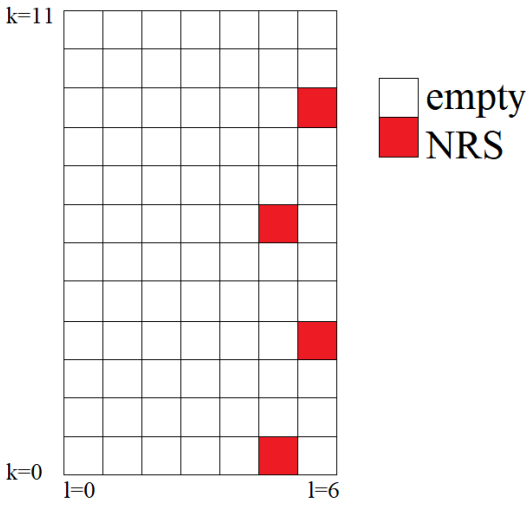


Figure 3.12: PRB only containing NRS in red.

Function	`referenceSignalGenerator`
Input	`nprimes - internally generated value`
	`Ncell - Cell ID`
	`m - norm defined variable`
	`l - last two OFDM symbol positons`
Output	`r - complex value output`

Table 3.10: referenceSignalGenerator function description.

Function	`referenceSignalFrame`
Input	`ns - slot number`
	`Ncell - Cell ID`
Output	`map - slot containing only NRS`

Table 3.11: referenceSignalFrame function description.

3.3.2 Narrowband Primary Synchronization Signal

The NPSS is generated in *narrowbandPrimarySynchronizationSignal.m*, described in table 3.12. The function needs no input values and outputs a full subframe containing the NPSS. The output subframe is as seen in figure 3.13.

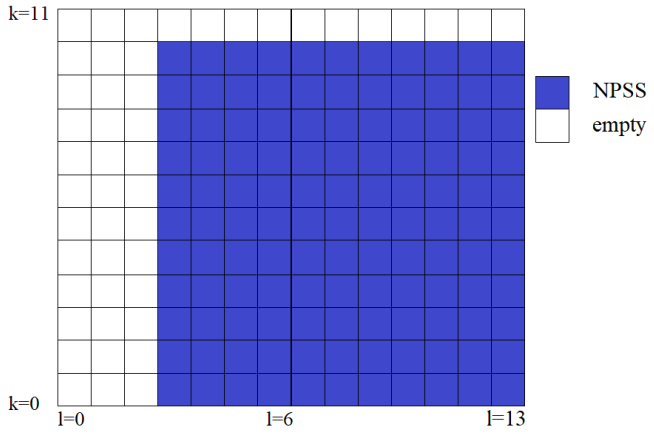


Figure 3.13: NPSS subframe .

The signal is generated based on a frequency domain length-11 Zadoff-Chu sequence.

Function	`narrowbandPrimarySynchronizationSignal`
Input	
Output	`sf5 - subframe containing the NPSS`

Table 3.12: narrowbandPrimarySynchronizationSignal function description.

3.3.3 Narrowband Secondary Synchronization Signal

The NSSS is generated in *narrowbandSecondarySynchronizationSignal.m*, described in table 3.13, receiving the Cell ID and the frame number as an input. The output is a subframe containing the NSSS as shown in figure 3.14.

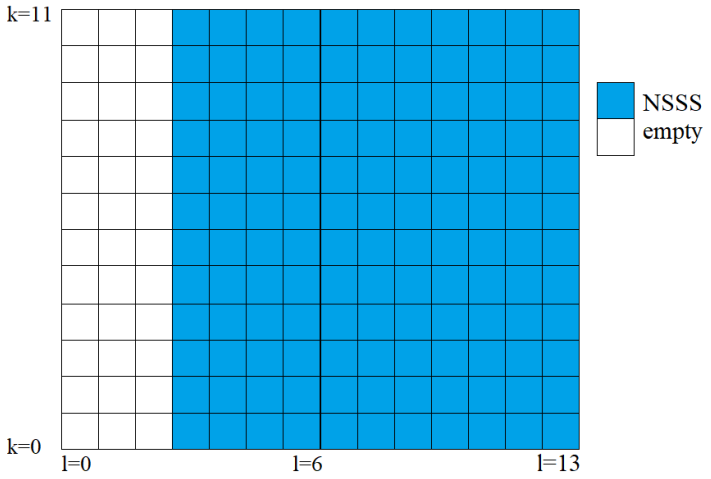


Figure 3.14: NSSS subframe.

The signal is generated based on a frequency domain length-131 Zadoff-Chu sequence.

Function	`narrowbandSecondarySynchronizationSignal`
Input	`nf - radio frame number`
	`Ncell - Cell ID`
Output	`sf9 - subframe containing the NSSS`

Table 3.13: narrowbandSecondarySynchronizationSignal function description.

3.4 Physical Channels Generation

The three physical downlink channels are individually generated using the above mentioned general procedures. The NPDSCH is generated in *downlinksharedChannelNPSCHGenerator.m*. The NPBCH is generated in *broadcastChannelNPBCHGenerator.m* and finally the NPDCCH is generated in *downlinkControlChannelNPDCCHGenerator.m* .

3.4.1 Narrowband Physical Broadcast Channel

The NPBCH is generated in script *broadcastChannelNPBCHGenerator.m*. This channel is responsible for carrying the MIB. The MIB values are set in *broadcastChannelNPBCHGenerator.m* in binary notation.

Figure 3.15 depicts the procedures for generating NPBCH samples implemented in *broadcastChannelNPBCHGenerator.m*.

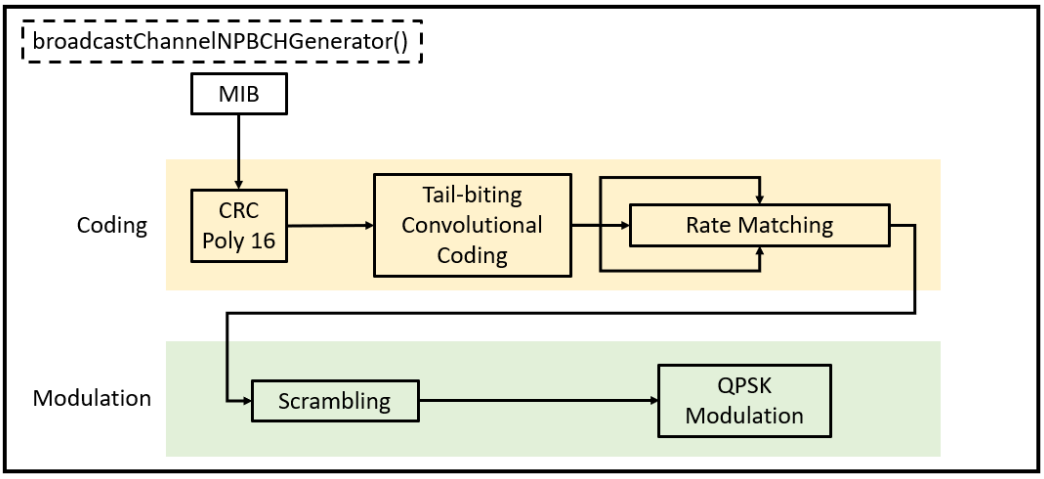


Figure 3.15: NPBCH samples generation procedure block diagrams.

To build the NPBCH the MIB is first concatenated with a lenght 16 CRC sequence generated of generator polynomial for one antenna port, defined in (3.5).

$$< 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 > \tag{3.5}$$

After the coding process the bits are then scrambled with initial condition defined in (3.6).

$$x2\_init = N_{ID}^{Cell} \tag{3.6}$$

The bits are afterwards modulated to QPSK symbols and mapped into a sub frame. The first 3 symbols are not used in the mapping process. The result is exemplified in figure 3.16.

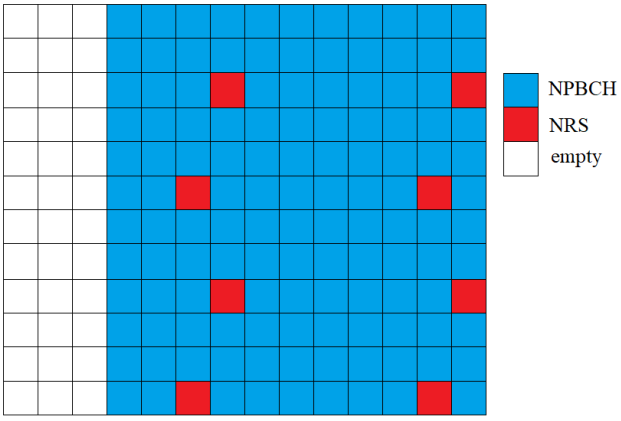


Figure 3.16: Generic example of an NPBCH generated sub frame.

The functions input and output is presented in table 3.14. To build the subframe with the NPBCH the function *mapperNPBCH* must be used and is described in table 3.15.

Function	broadcastChannelNPBCHGenerator
Input	rv_idx - revolution number
	Ncell - Cell ID
Output	npbch - NPBCH complex values

Table 3.14: broadcastChannelNPBCHGenerator function description.

Function	mapperNPBCH
Input	npbch - NPBCH complex values
	map - slot containing NRS
Output	sf0 - NPBCH subframe

Table 3.15: mapperNPBCH function description.

3.4.2 Narrowband Physical Downlink Control Channel

The NPDCCH is generated in script *downlinkControlChannelNPDCCHGenerator.m*. This is the channel responsible for carrying DCI.

The data is set in the function aswell as the possible DCI formats (N0, N1 and N2). The data is set element by element in binary format and then concatenated into a bitstream to enter the coding and modulation chain.

The generation of the NPDCCH samples generated in *downlinkControlChannelNPDCCH-Generator.m* is shown in figure 3.17.

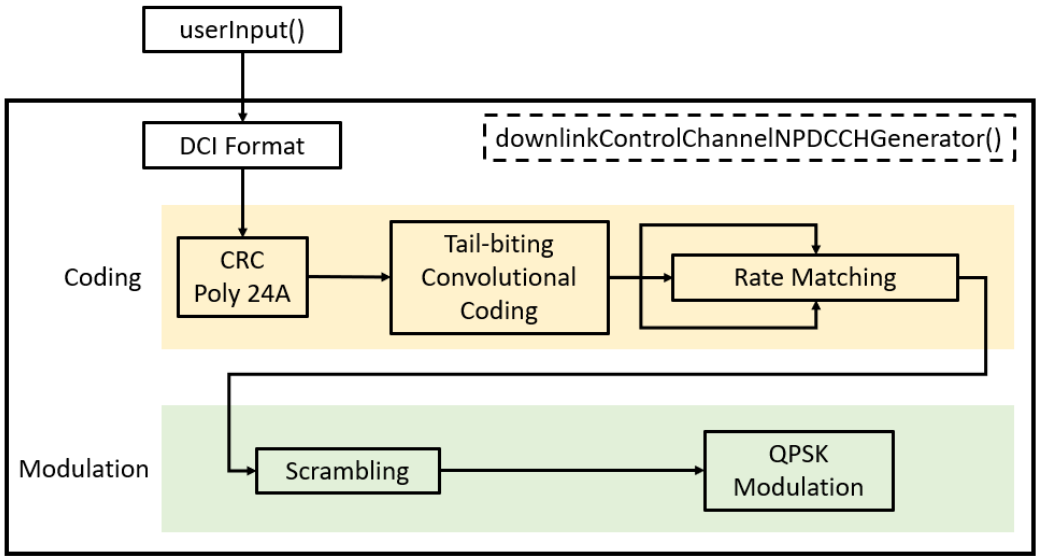


Figure 3.17: NPDCCH samples generation procedure block diagrams.

After data is created and selected the coding process begins.

The data bitstream is first appended with CRC sequence bits generated with the polynomial 24A. Tail-biting convolutional coding and rate matching is then applied.

The resulting bits are scrambled using initial condition defined in (3.7).

$$x2\_init = \lfloor n_s/2 \rfloor \, 2^9 + N_{ID}^{Cell} \tag{3.7}$$

The scrambled bits are modulated into QPSK symbols.

The modulated symbols are then mapped into PRB, the resource elements are filled according to the NPDCCH desired format. NPDCCH has two possible formats as shown in table 3.16. These formats define the number of Narrowband Control Channel Element (NCCE) to use in the same subframe.

NPDCCH Format	Number of NCCE
0	1
1	2

Table 3.16: NPDCCH Formats and respective number of NCCE.

The NCCE 0 occupies subcarriers 0 to 5 and NCCE 1 subcarriers 6 to 11 as shown in figure 3.18:

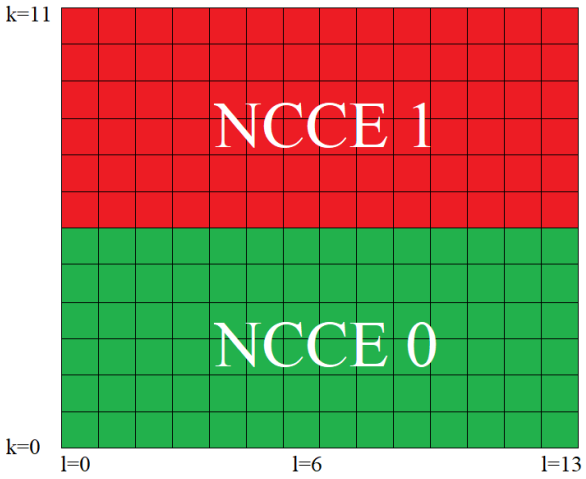


Figure 3.18: NCCE distribution for subframe.

After the NCCE format is chosen, the resource elements begin being filled. Resource elements already filled with NRS values will not be used.

To generate the NPDCCH complex values the used function is described in table 3.17. To build the NPDCCH subframe the function in table 3.18.

Function	`downlinkControlChannelNPDCCHGenerator`
Input	`Ncell - Cell ID`
	`ns - slot number`
	`rv_idx - revolution number`
Output	`dci - DCI content bitstream`
	`format - DCI format number`
	`npdcch - NPDCCH complex values`

Table 3.17: broadcastChannelNPBCHGenerator function description.

Function	mapperNPDCCH
Input	npdcch - NPDCCH complex values
	map - slot containing NRS
Output	sfc - NPDCCH subframe

Table 3.18: mapperNPDCCH function description.

3.4.3 Narrowband Physical Downlink Shared Channel

The NPDSCH is generated in script *downlinksharedChannelNPSCHGenerator.m*. The data is set as input variable x to then enter the coding and modulation chain. The data is controlled by the scheduling information of the NPDCCH format N1 message.

The overall procedure is shown in figure 3.19.

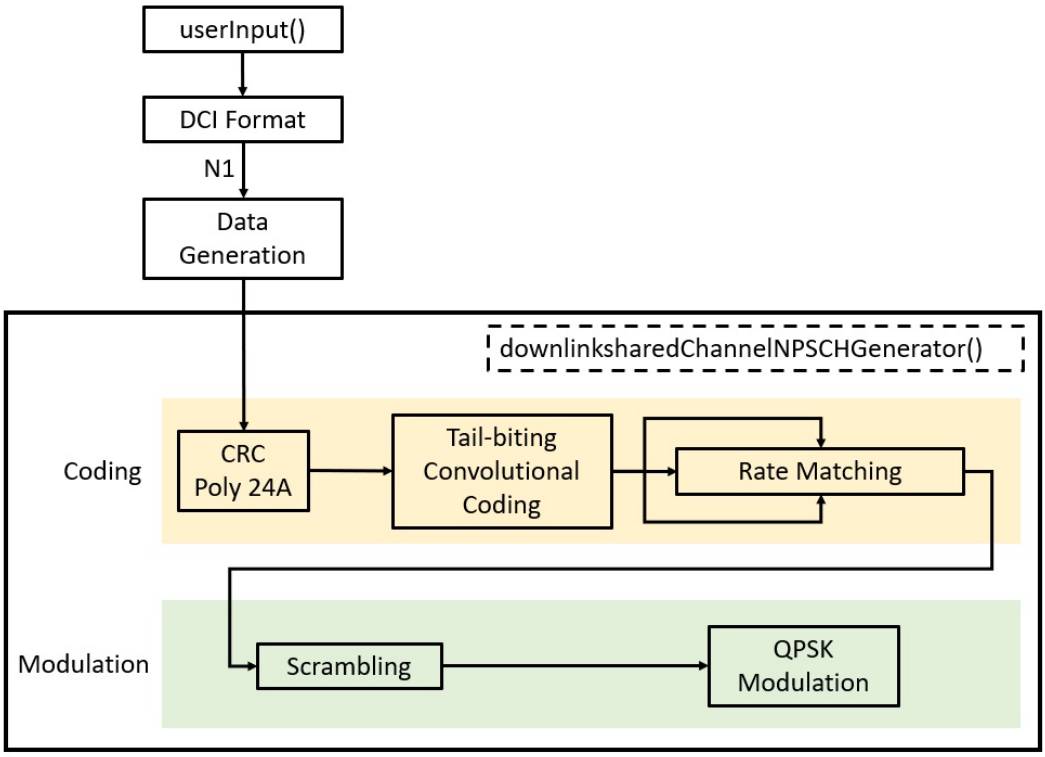


Figure 3.19: NPDSCH samples generation procedure block diagrams.

The data is then appended with a CRC sequence generated with polynomial 24A. The bits are then coded in the Tail-biting Convolutional enconder and rate matched.

Afterwards the codeword is scrambled with initial condition presented in (3.8).

$$x2\_init = n_{RNTI} \cdot 2^{14} + n_f \mod 2 \cdot 2^{13} + \lfloor n_s/2 \rfloor 2^9 + N_{ID}^{Cell} \tag{3.8}$$

The scrambled codeword is modulated to QPSK symbols.

The resource elements within a subframe are then filled if not filled with NRS elements already.

The NPDSCH can occupy one or more subframes depending on the repetition number (MNPDSCHrep given by the DCI paramater N_{Rep}) and the Broadcast Channel (BCCH) flag.

If the NPDSCH is carrying the BCCH the flag is set to '1'. In this case, the subframes containing the NPDSCH are filled continuously and then repeated MNPDSCHrep times.

If the NPDSCH does not contain the BCCH then the flag is set to '0'. Every subframe filled is then repeated min(MNPDSCHrep,4)-1 times before moving on to filling the remaining subframes. The repetitions are examplified in figures 3.20 and 3.21.

Figure 3.20: NPDSCH subframes containing BCCH repetition example

Figure 3.21: NPDSCH subframes not containing BCCH repetition example.

To generate the complex values for the NPDSCH the function in table 3.19 is called. To build the subframes for the NPDSCH the function in table 3.20 must be used.

Function	`downlinksharedChannelNPSCHGenerator`
Input	Ncell - Cell ID
	`ns - slot number`
	`rv_idx - revolution number`
	`dci - DCI content bitstream`
	`x - DL-SCH data`
	`nf - frame number`
Output	`npdsch`

Table 3.19: downlinksharedChannelNPSCHGenerator function description.

Function	`mapperNPDSCH`
Input	`npdsch - NPDSCH complex values`
	`MNPDSCHrep - repetition number`
	`map - slot containing NRS`
	`flagBCCH - flag indicating if SIB is being carried`
Output	`stot - NPDSCH subframes`

Table 3.20: mapperNPDSCH function description.

3.5 Scheduling

The scheduling procedure intents to form a radio frame, joining together all the previously generated channels and signals.

3.5.1 NPSS and NSSS

The NPSS is always transmitted in subframe 5 [3GP16].

The NSSS is transmitted in even number radio frames in subframe 9. The NSSS varies according to nf, radio frame number, with cyclic shift given by:

$$\Theta_f = \frac{33}{132}(n_f/2) mod 4 \tag{3.9}$$

3.5.2 NPBCH

The NPBCH is transmitted in subframe 0 of every radio frame. The Time Transmission Interval (TTI) for the NPBCH is of 640ms meaning it is transmitted over the span of 64 radio frames before being update with the new system frame number and parameters that might have changed. Figure 3.22 shows an example of a scheduled radio frame for the synchronization signals and the NPBCH. [Sha17]

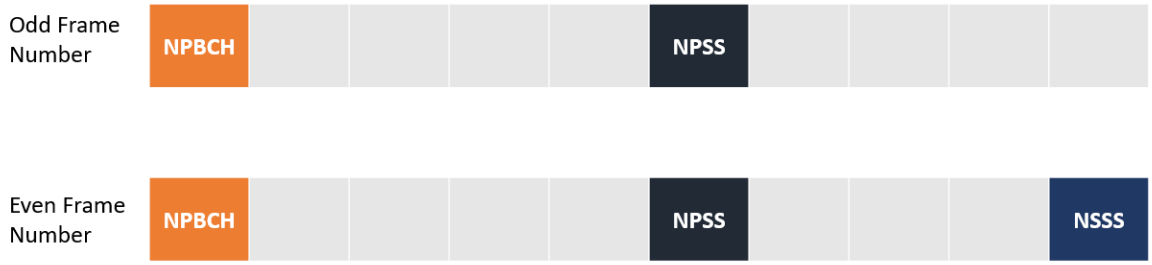


Figure 3.22: Radio Frame scheduling example for the NPSS, NSSS and NPBCH.

3.5.3 Implemented Scheduler

The implemented scheduler places the NPBCH in the first subframe, subframe 0, the NPSS in subframe 5 and NSSS in subframe 9, for every odd frame number. The remainder subframes shall de filled with both NPDSCH and NPDCCH subframes.

The channels are scheduled in a dynamic way occupying different subframes regarding the current procedure.

The implemented scheduler only builds one radio frame. The scheduling is compliant with the standard although limited and incomplete. The NPDCCH is always scheduled for subframe 1. If the NPDCCH carries the DCI format N1 message to schedule the NPDSCH, the NPDSCH is always 4 subframes apart from the NPDCCH [3GP17b] subframe issuing it, an example can be seen in figure 3.23.

The SIB1-N has a fixed scheduled and is always set for subframe 4. It is then transmitted and repeated in every other radio frame. The minimum TBS value for SIB1-NB transmission is 208 bits which translates into three subframes. Since the implemented generation function

only outputs one subframe then only one part of the SIB1-N is transmitted. The SIB1-N transmission is presented in figure 3.24.

Figure 3.23: Example of resulting Radio Frame by the implemented scheduler.

Figure 3.24: Example of resulting Radio Frame by the implemented scheduler with SIB1-NB transmission.

3.6 OFDM Signal Generation

The OFDM signal generation procedure is made in three steps symbol wise.

Every symbol in the previously generated radio frame is of 12 elements, so in preparation for the Inverse Fast Fourier Transform (IFFT) the number must be raised to the closest power of 2, hence zero padding is applied. To avoid the DC carrier 4 zeros are padded between carriers 6 and 7, raising the number of elements to 16, as shown in figure 3.25.

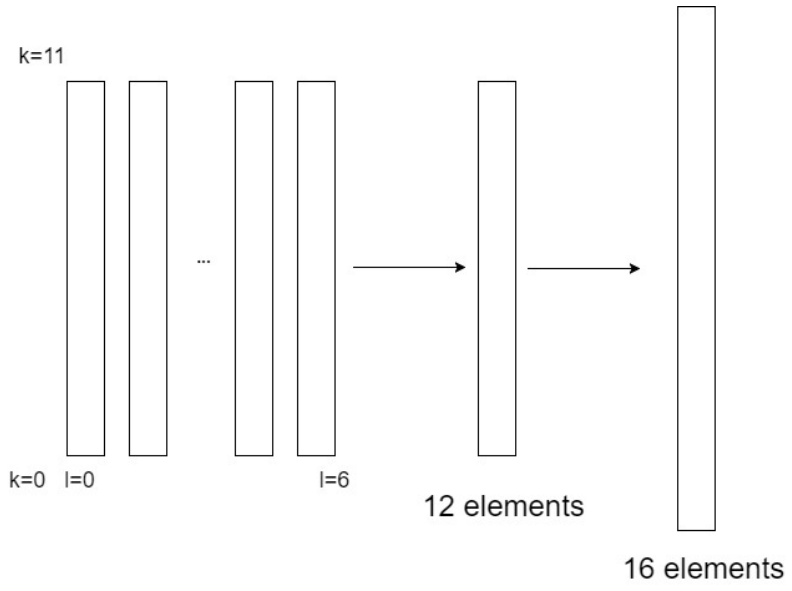


Figure 3.25: Symbol isolation and zero padding procedure.

The IFFT is then applied to convert the symbols from frequency to time domain, as shown in figure 3.26.

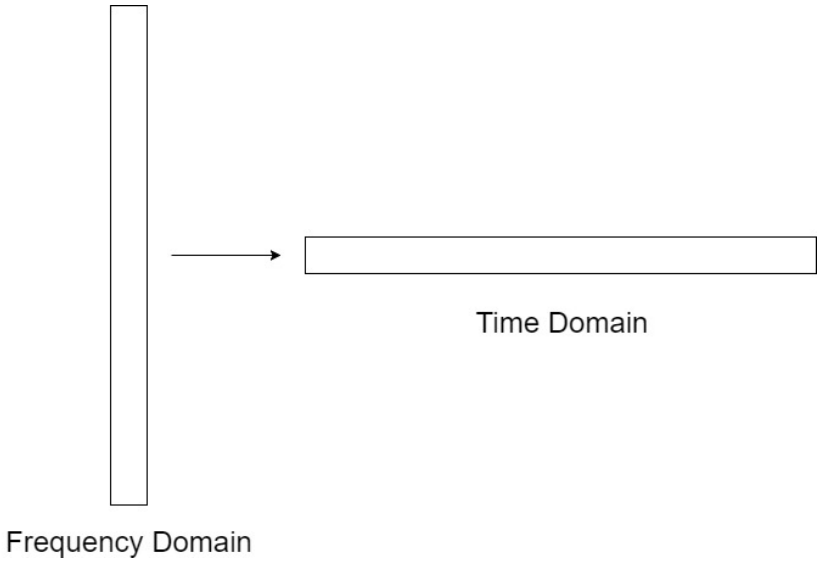


Figure 3.26: Frequency to Time domain conversion using IFFT.

The time domain samples are then upsampled by interpolation with a value of 8 resulting in the desired 128 subcarriers. This procedure is needed to ensure the 1.92MHz sampling rate, as shown in figure 3.27.

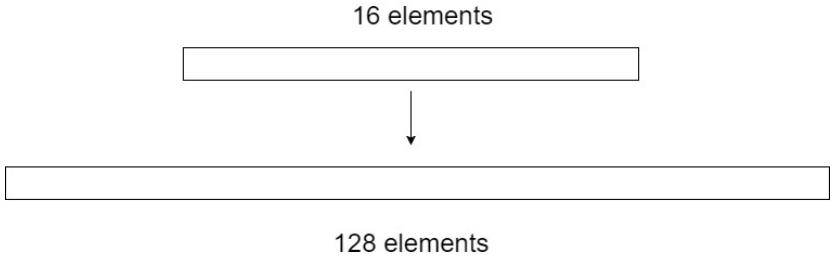


Figure 3.27: Upsampling Procedure.

Finally, to reduce Inter-Symbol Interference (ISI), Cyclic Prefix (CP) is applied. CP is added by copying the last symbols in the carrier and pasting them before the first symbols. In NB-IoT the CP length varies between 10 and 9 according to the symbols position in the PRB, as shown in figure 3.28.

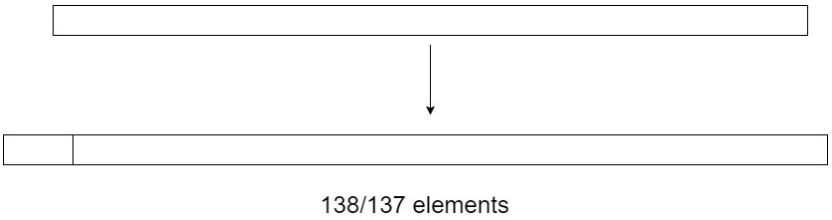


Figure 3.28: CP insertion.

The CP is added in *cpInsertion.m*

Function	`cpInsertion`
Input	`s1 - Upsampled time samples`
Output	`symbolsToSend - NB-IoT downlink baseband samples`

Table 3.21: cpInsertion function description.

On the output of these procedures are standard compliant baseband samples for NB-IoT on one antenna port in standalone mode.

The implementation for the transmitter side is capable of performing all procedures (CRC, Encoding, Rate Matching, Scrambling, Modulation and OFDM signal generation) compliant to 3GPP's release 13.

All channels are generated standard compliant although the scheduling procedures for the NPDSCH are incomplete.

The data transported in the generated baseband samples can be recovered in the implemented receiver which is described in the next chapter.

Chapter 4

Matlab Modelling of the Downlink Receiver

This chapter presents the Matlab modelling and implementation of the NB-IoT downlink receiver. The implemented chain and procedures are demonstrated.

4.1 Introduction

The Receiver side, responsible for decoding and demodulating the transmitted data is implemented in *recoverDLsignal.m* . The script receives the data as input and performs various functions to deliver data to the uplink device. The procedures are shown in figure 4.1

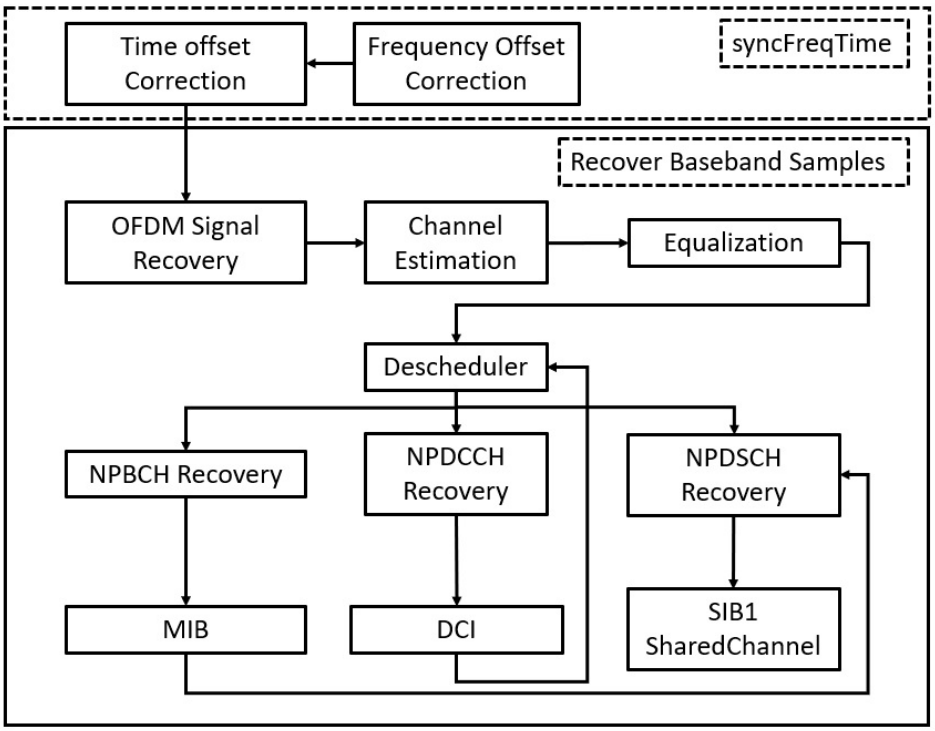


Figure 4.1: Implemented receiver macro block diagram.

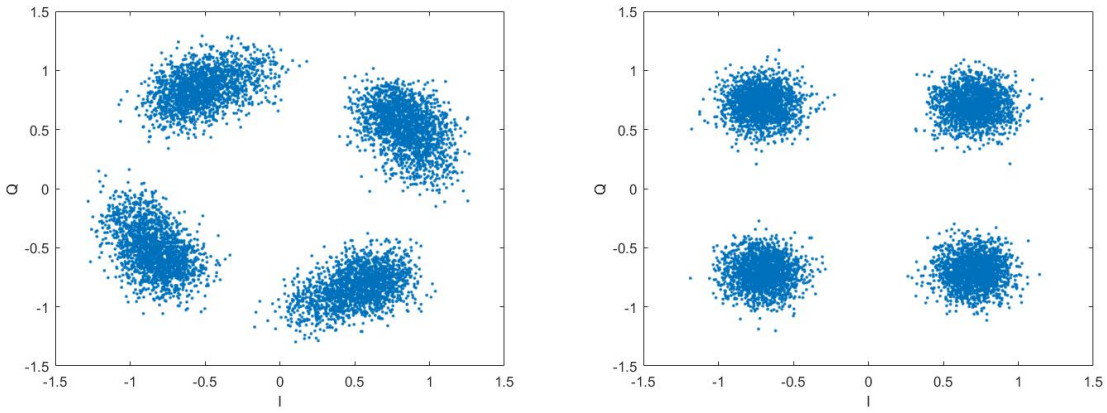
4.2 Frame Correction

Equipment differences between transmitter and receiver devices can deteriorate the signal. Oscillator differences and finite capture windows may jeopardize the signal's correct reception. To mitigate these effects frame correction procedures were developed.

The receiver side needs to be equipped with tools to synchronize the received frame, compensating for occasional local oscillator differences and time symbol offset due to the finite receiver window.

4.2.1 Frequency Compensation

Carrier Frequency Offset (CFO) is a common mismatch in communication systems, which can occur due to transmitter and receiver devices differences on local oscillators, Doppler frequency shifts and others. The consequences of this mismatch can be demonstrated in figures 4.2 a) and b).



(a) Constelation diagram of QPSK symbols before correcting CFO.

(b) Constelation diagram of post CFO corrected QPSK symbols.

Figure 4.2: Example of CFO damage and correction.

Frequency compensation is applied to correct CFO. CFO must first be estimated, this procedure is done in $FrequencyOffset.m$. The function is used as described in table 4.1.

Function	`FrequencyOffset`
Input	`waveform - received signal`
	`Nfft - number of subcarriers`
	`deltaf - subcarrier separation in Hz`
	`sr - sampling rate`
	`cpsfvalue - CP samples repeated per symbol in a subframe`
Output	`delta_f - frequency offset`

Table 4.1: FrequencyOffset function description.

The estimation is done by comparing the angle difference between the received OFDM symbols' CP and the symbols from which the CP was copied [Mat].

The CFO can be expressed in time domain as a frequency shift, as translated in (4.1).

$$y(t) = x(t)e^{j2\pi f_{offset}t} \tag{4.1}$$

With this in mind, knowing the estimated value for f_{offset}, the correction can be done by multiplying the received signal with the symmetrical shift, which translates in (4.2).

$$x(t) = y(t)e^{-j2\pi f_{offset}t} \tag{4.2}$$

This procedure is done in *FrequencyCorrect.m*, described in table 4.2.

Function	`FrequencyCorrect`
Input	`in - received signal`
	`foffset - frequency offset value`
	`sr - sampling rate`
Output	`out - frequency corrected symbol`

Table 4.2: FrequencyCorrect function description.

4.2.2 Time Symbol Offset Correction

Time symbol offset is an error that compromises the synchronization for OFDM symbols.

The effect can occur when the transmitter and receiver device have different time references and when finite capture windows are used.

The OFDM reception is jeopardized in the sense that the expected symbols are shifted in time thus their demodulation and CP removal will be wrongly performed. [WS95]

The captured samples may be shifted of their ideal place due to the capture window being finite, as shown in figure 4.3 , so when receiving a frame time synchronization must be applied.

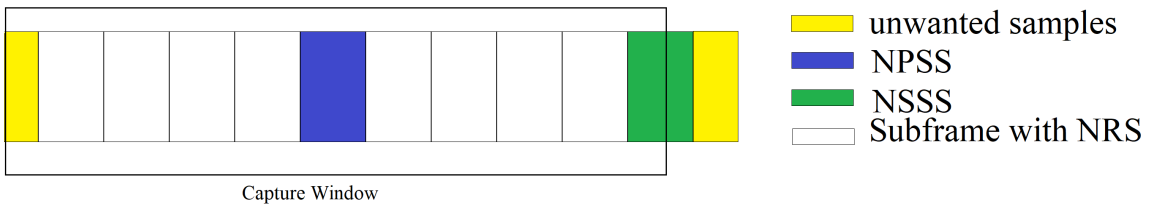


Figure 4.3: Demonstration of frame offset error.

The time offset is first detected in *DLFrameOffset.m*, table 4.3. This procedure is made by generating a reference frame with only the resources elements with NRS and NPSS are filled, the remaining are set to zero.

The input signal is then correlated with the reference frame to detect peak values thus calculating the number of shifted/lost samples [Mat17c].

Function	DLFrameOffset
Input	waveform - frequency corrected received signal
	cpsfvalue - CP samples repeated per symbol in a subframe
	sr - sampling rate
	out - frequency corrected symbol
	ns - slot number
	Ncell - Cell ID
Output	offset - time symbol offset

Table 4.3: DLFrameOffset function description.

The correction is made by removing the unwanted samples and filling the rest of the frame with zeros. Even though some information might get lost it's compensated due to the radio frames repetitions.

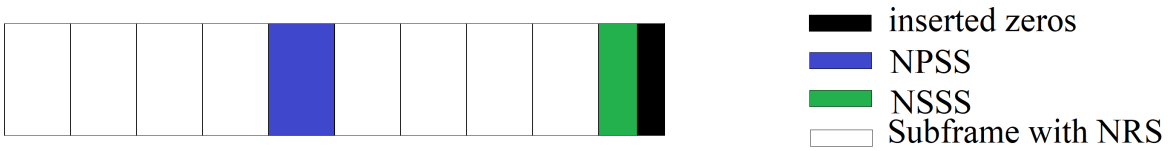


Figure 4.4: Demonstration of frame offset correction and zero insertion.

4.3 OFDM Demodulation

The OFDM signal demodulation begins with CP removal which is done by extracting the correct symbols from the received signal ignoring the CP patch, as depicted in figure 4.5.

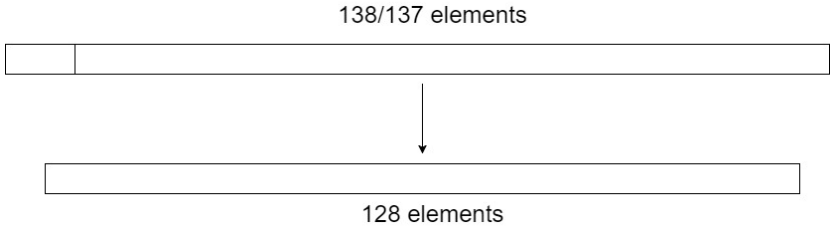


Figure 4.5: CP removal.

After CP removal a resampling is made to remove the interpolation made, thus reducing the number of symbols by a factor of 8, as illustrated in figure 4.6.

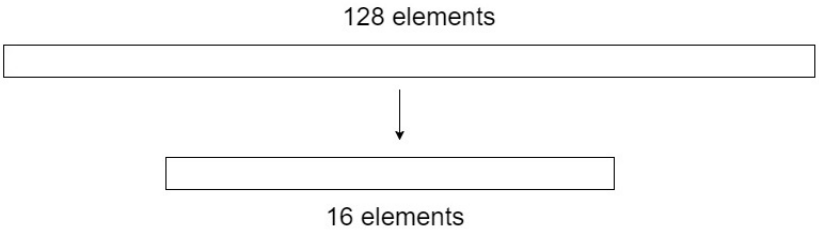


Figure 4.6: Downsampling procedure.

Fast Fourier Transform (FFT) is then applied to convert the samples back to frequency domain, as shown in figure 4.7.

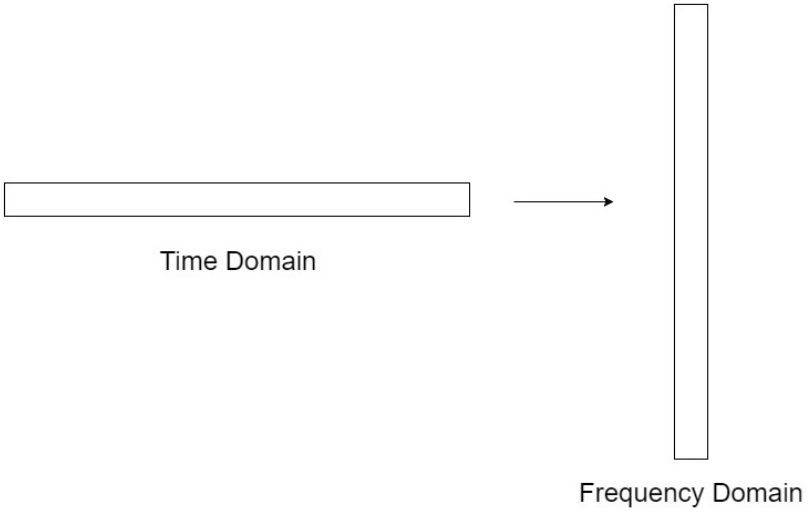


Figure 4.7: FFT application.

With the frequency domain samples zero unpadding can now be applied, removing the zeros inserted between subcarries 6 and 7, as shown in figure 4.8. The grid is then rebuilt.

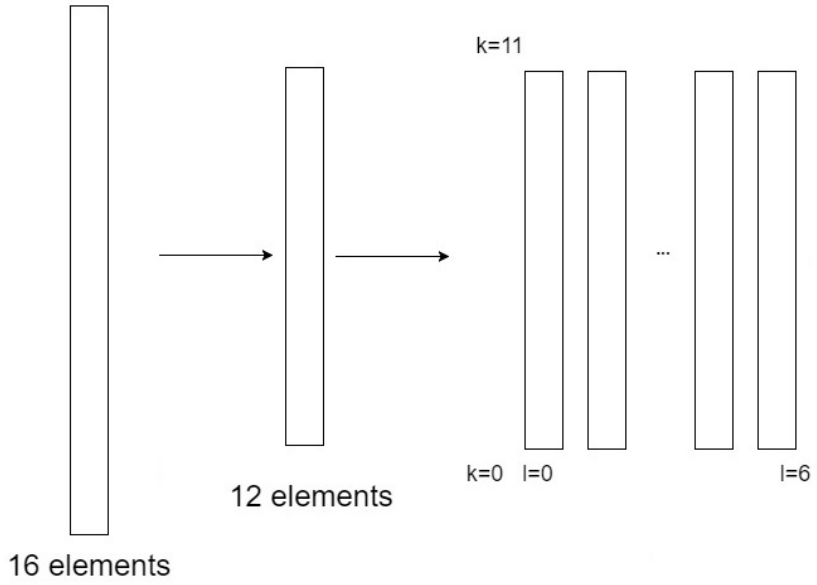


Figure 4.8: Zero Unpadding and grid rebuilding.

This procedure is implemented in *recoverDLsignal.m* inline with the remaining code.

4.4 Equalization

The equalization procedure intends to reverse the signal's distortion that occured through the channel.

Zero Forcing (Z-F) was the method chosen, it works by forcing the signal to a zero ISI state applying the reciprocal value of the channel effect.

The procedure starts by estimating the channel. The effects applied due to the channel can be translated into equation (4.3)

$$Y = HX + N \tag{4.3}$$

Where Y is the received signal, X the transmitted signal, H the channel effect ad N the Additive White Gaussian Noise (AWGN).In this case, since we're using Z-F, the channel approximation is given by (4.4).

$$Y = H_{ZF}X \tag{4.4}$$

The equation now approximates the received signal Y, to the product between the transmitted signal X and the Z-F channel estimate H_{ZF}. To which follows (4.5).

$$H_{ZF} = \frac{Y}{X} \tag{4.5}$$

This calculation is made thanks to the NRS, by comparing the received NRS values with the expected transmitted values.

The estimation procedure is performed in *downlinkChannelEstimate.m*. The function receives as input a received subframe and a reference grid which is an empty subframe containing only the NRS in NRS defined positions.

Function	`downlinkChannelEstimate`
Input	`rxgrid - received signal's grid`
	`refgrid - reference grid with NRS only`
Output	`H_EST - channel estimate`

Table 4.4: downlinkChannelEstimate function description. rxGrid is the received subframe and refGrid a locally generated subframe containing only NRS values.

The NRS is only present in four subcarriers, hence to estimate the channel for all subframes the interpolation method is used.

The channel estimates in a subframe are first averaged in the four subcarriers with NRS subcarriers as shown in figure 4.9.

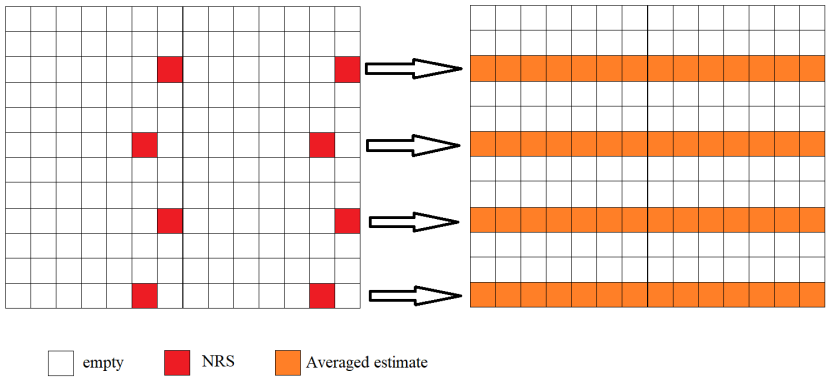


Figure 4.9: Channel estimates in NRS resource elements on the left, averaging for channel estimate in NRS subcarriers on the right.

To obtain the channel estimate for the remaining subcarriers linear interpolation is used. With this method a channel estimate is determined for each subcarrier as shown in 4.10: [Mat17a]

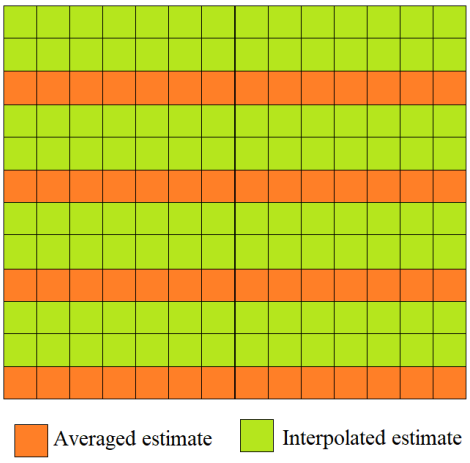


Figure 4.10: Interpolation for channel estimate calculation on non NRS subcarriers.

The equalization process is done in *equalizer.m* which applies the channel estimate value to the received signal to approximate it to the transmitted signal. The process can be explained by equations (4.6) and (4.7).

$$\frac{Y}{H_{ZF}} = \frac{XH_{ZF}}{H_{ZF}} \tag{4.6}$$

$$\frac{Y}{H_{ZF}} = X \tag{4.7}$$

Function	`equalizer`
Input	`rxgrid - received signal's grid`
	`H_EST - channel estimate`
Output	`out - equalized signal`

Table 4.5: equalizer function description.

4.5 General Demodulation and Decoding Procedures

After correcting the offsets of the received signal and recovering it back to frequency the channels now enter the demodulation and decoding procedures which are outlined in figure 4.11.

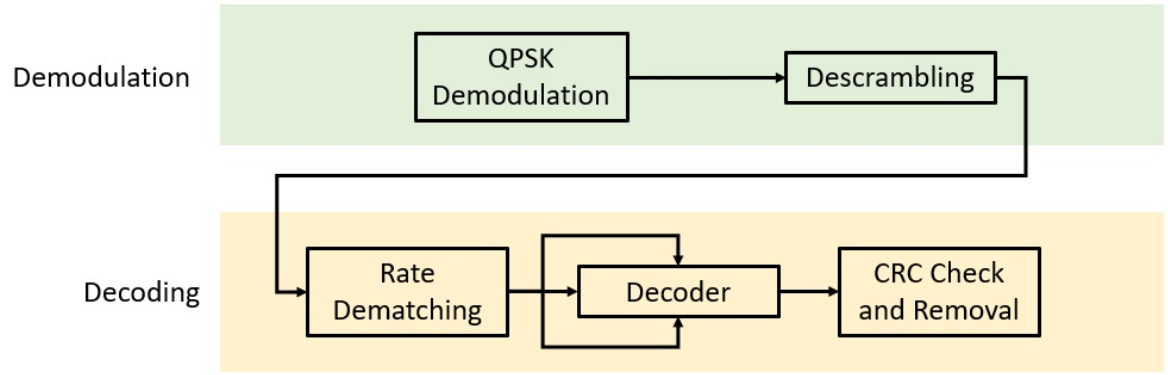


Figure 4.11: Demodulation and Decoding procedures.

4.5.1 Demodulation

To obtain the received bits and corresponding data the script must orderly extract the bits from the PRB ignoring the NRS resource elements. The obtained bistream is then applied to the demodulation and decoding procedures.

4.5.1.1 NRS Extraction

The NRS occupies fixed places in the PRB as shown in figure 3.12. The script *element-demapper.m* reshapes the PRB to a line and sweeps every position removing those containing the NRS. The procedure is represented in figure 4.12.

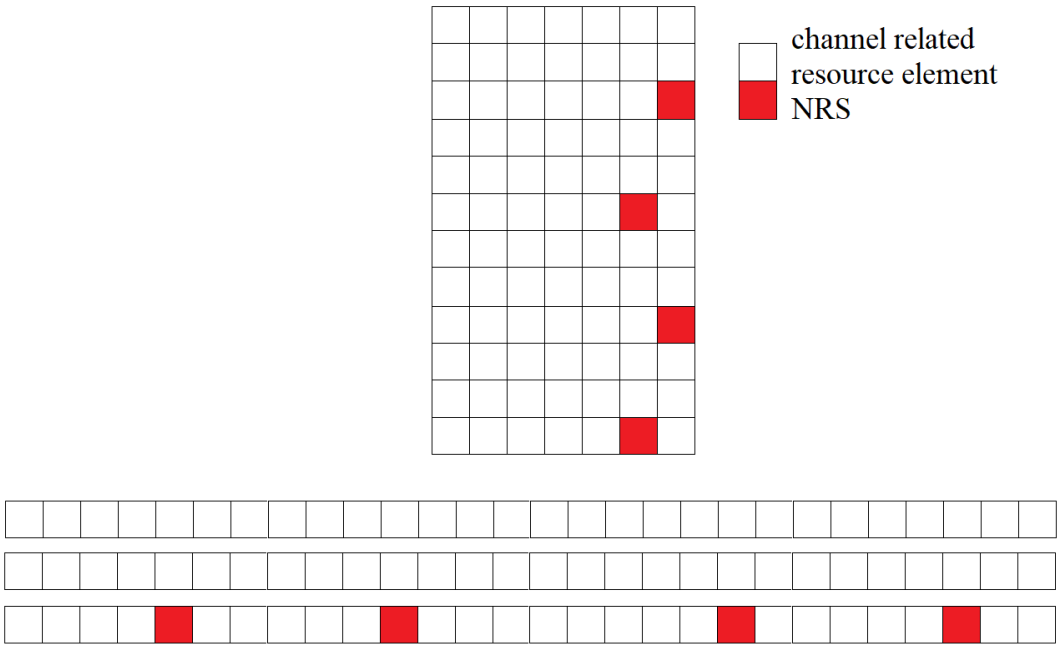


Figure 4.12: Reshaping the PRB into a line.

Function	`elementdemapper`
Input	`input - input channel`
Output	`line - channel without NRS`

Table 4.6: elementdemapperfunction description.

Using this method facilitates the sweeping process and also provides an output in line form which eases the inputs for the remaining functions.

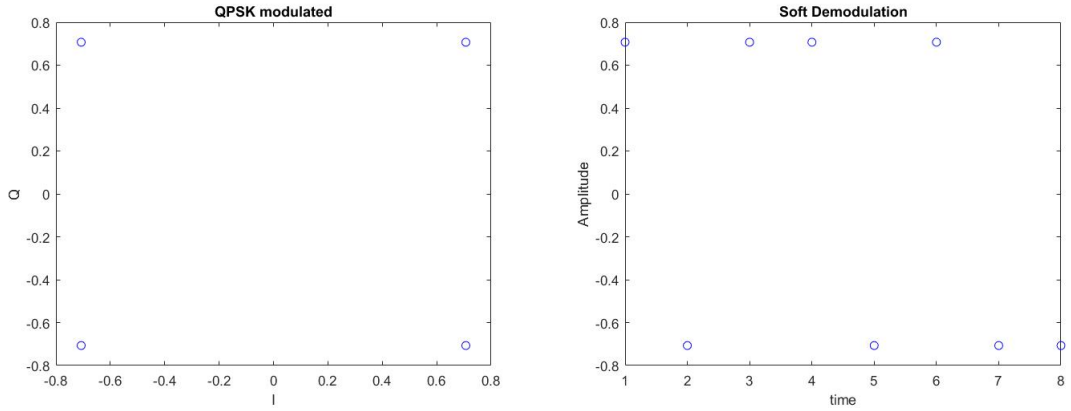
4.5.1.2 QPSK Demodulation

The demodulation process, of converting complex modulated values back into bits, is performed in function *demod_data.m*. This function takes as input the data to demodulate, the modulation scheme (2 for QPSK) and the type of decision (0 for hard decision and 1 for soft decision).

The chosen decision was soft decision for every channel, an example of the output is shown in figure 4.13.

Function	`demod_data`
Input	`data_symbol - modulated symbol`
	`Qm - modulation order`
	`coder_type_value - flag for soft or hard decision`
Output	`decoded_data`

Table 4.7: demod_data description.



(a) Constelation diagram of QPSK symbols. (b) QPSK symbols demodulated using soft decision.

Figure 4.13: Example of QPSK modulation and soft decision demodulation.

4.5.1.3 Descrambling

The descrambling procedure is similar to the scrambling procedure, it uses the same known initial condition to generate the gold sequence in *goldseq.m* and then performs an *XOR* to the input data. The procedure is implemented in *descramble.m* for the NPDSCH, *descrambleControl* for the NPDCCH and *descrambleNPBCH* for the NPBCH. The difference between them is the gold sequence initialization set inside. The function descripition is presented in table 4.8.

Function	`descramble`
Input	`data_in -soft log data`
	`- data necessary for the scrambling sequence for each channel`
Output	`data - descrambled data`

Table 4.8: descramble function description.

4.5.2 Decoding

The decoding procedure holds the intent of converting a codeword back to the original data bitstream.

4.5.2.1 Rate Dematching

The Rate Dematching procedure was designed in complete simmetry to the Rate Matching procedure. It starts by reordering the bits according to the puncturing pattern, this step is done in function *depuncturer.m*.

Function	depuncturer
Input	outsize - desired output length
	input
Output	depunctered

Table 4.9: depuncturer function description.

To deinterleave the bitstream an interleaved stream of zeros is generated, since some bits might have been removed or $< NULL >$ bits inserted. The depuncturer's output is then placed in the helper stream in the available places, i.e., non $< NULL >$ bits. The procedure is shown in 4.14.

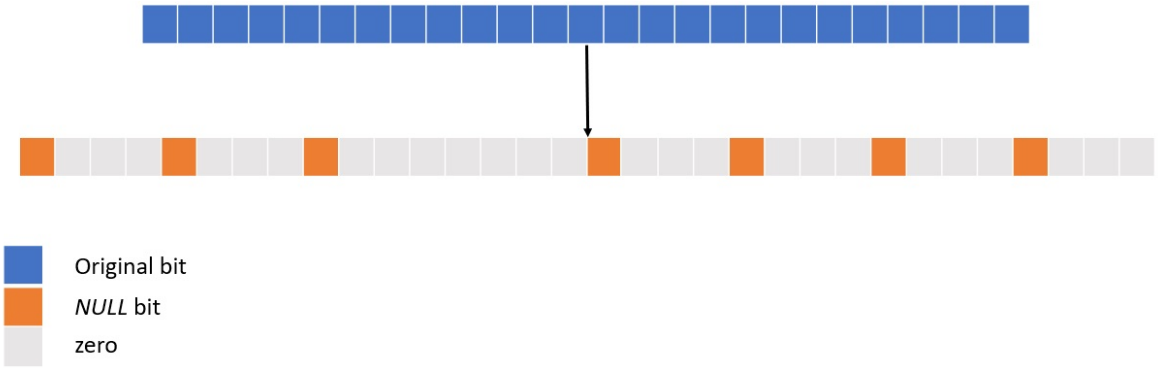


Figure 4.14: Concatenation of input bits with helper interleaved matrix.

The resulting bitstream is then divided into three bitstreams accordingly, to be deinterleaved. The deinterleaving procedure is made in *subblock_deinterleaver.m*, described in table 4.10, by applying the reverse permutation of the matrix. The deinterleaved output is then filered to remove the inserted $< NULL >$ bits by the helper matrix.

The procedure can be explained in figure 4.15 and its full implementation is made in *rateDeMatching.m*, described in table 4.11.

Function	subblock_deinterleaver
Input	d - input data
	superscript - input data index
Output	v - deinterleaved data

Table 4.10: subblock_deinterleaver function description.

Function	rateDeMatching
Input	e - input data
	outlen - desired output length
Output	a0 - systematic data
	a1 - parity1 data
	a2 - parity2 data

Table 4.11: rateDeMatching function description.

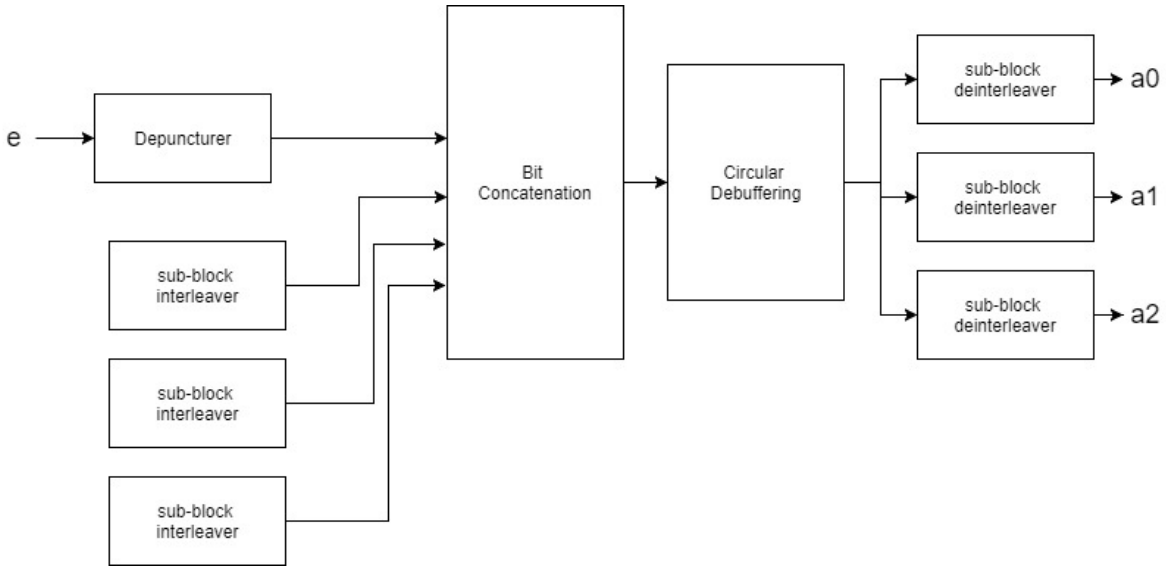


Figure 4.15: Rate dematching procedure using auxiliary interleavers.

4.5.2.2 Tail Biting Decoding

To the three rate dematching outputs turbo decoding is applied to decode the tail biting convolutionally encoded codewords.

To reduce delays on the project development this procedure is excepcionally performed by a Matlab Communications toolbox function, *lteConvolutionalDecode.m*.

Function	lteConvolutionalDecode
Input	arg - input data, concatenated array of the dematched outputs
Output	out - decoded data

Table 4.12: lteConvolutionalDecode function description. arg is the input of the concatenated outputs of the rateDeMatching procedure [a0 a1 a2] [Mat17b].

4.5.2.3 CRC Check

CRC was first applied to the transmitted bitstream to support forward error correction. The function *checkcrc.m* verifies the presence of errors.

The procedure is made by performing *XOR*s between the input bitstream and the polynomial initially used to append the CRC bits.

It returns a flag indicating '1' for the presence of errors or '0' otherwise.

Function	`checkcrc`
Input	`input - data with appended sequence`
	`poly - generator polynomial`
Output	`error - indication flag`

Table 4.13: checkcrc function description.

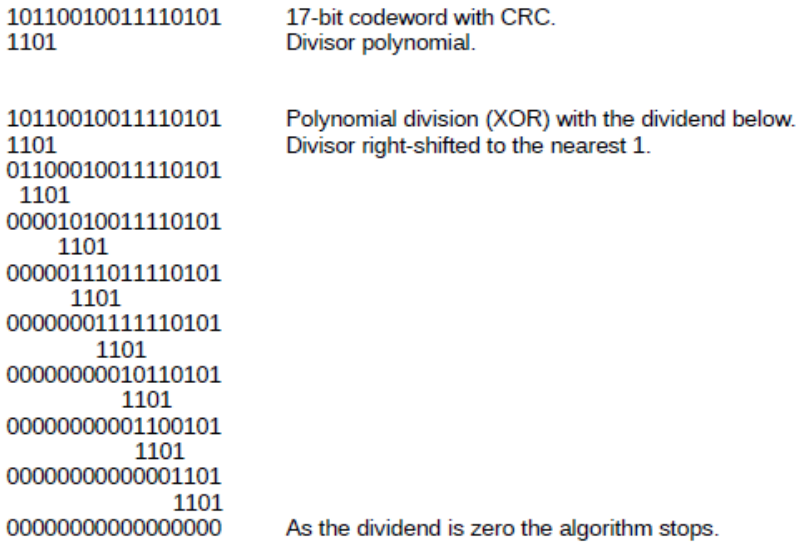


Figure 4.16: Example of CRC error detection [TK17].

4.6 Channel Extraction and Data Recovery

The descheduling procedure is done by sweeping through the full radio frame and dividing it in subframes, distributing the subframes to the proper variables. After the descheduling procedure the receiver side now holds the variables containing the NPBCH, NPSS, NSSS, NPDSCH and NPDCCH.

4.6.1 Descheduling

The descheduling procedure begins after receiving the radio frame and having it organized. The NPBCH, NPSS and NSSS are immediately extracted as shown in figure 4.17.

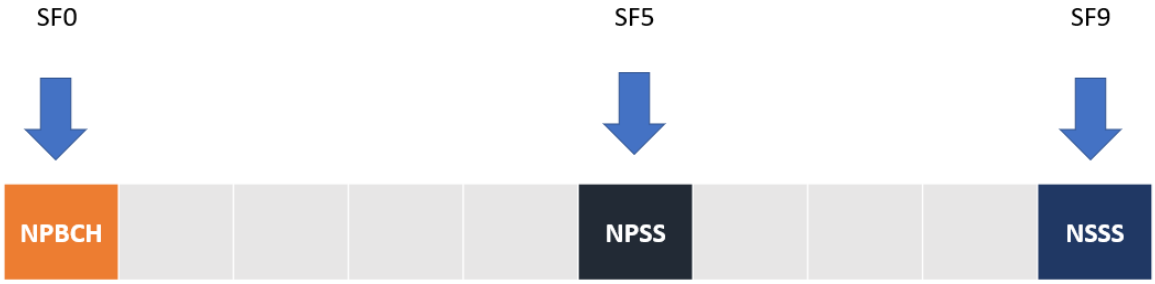


Figure 4.17: Descheduling procedure, extraction of the NPBCH, NPSS and NSSS subframes.

Afterwards the descheduler starts sweeping the remaining subframes. Once a NPDCCH subframe carrying the N1 DCI format is found then the descheduler assumes that 4 subframes ahead there's a NPDSCH subframe. The process is shown in figure 4.18.

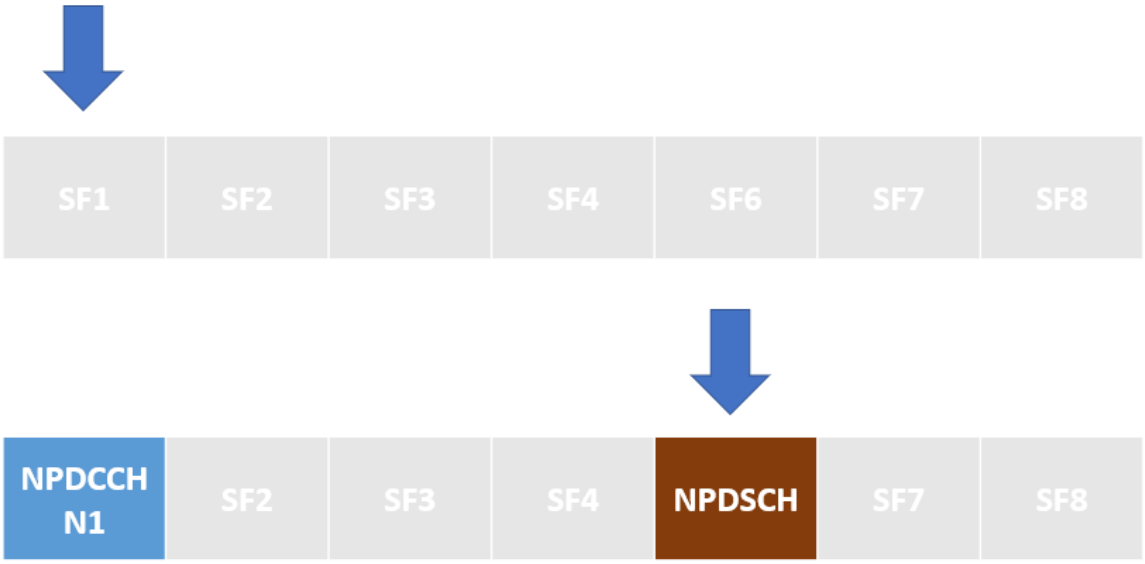


Figure 4.18: Descheduling procedure, identifying the NPDCCH and NPDSCH subframes.

When SIB1-N is scheduled, it has a fixed schedule for subframe 4 and is again easily extracted. The extracted MIB from the NPBCH provides the scheduling information regarding TBS for proper demodulation of the SIB1-N part. The procedure is shown in figure 4.19.

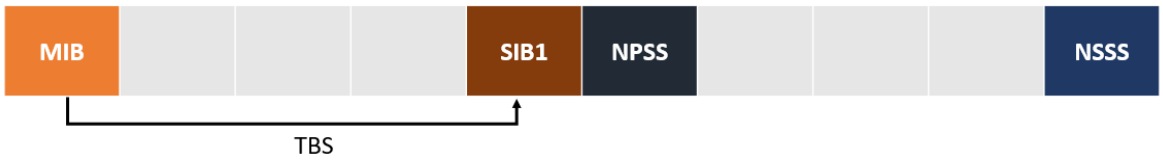


Figure 4.19: Descheduling procedure, identifying the TBS for SIB1-NB extraction.

4.6.2 NPBCH

The NPBCH is in subframe 0 of every radio frame. This makes it easy to identify the subframe holding the NPBCH.

The subframe goes through the general demodulation procedures delivering the NPBCH codeword. The codeword is the concatenation of the MIB bits which can now be easily extracted and delivered to the user.

An overview of the procedure is shown in figure 4.20.

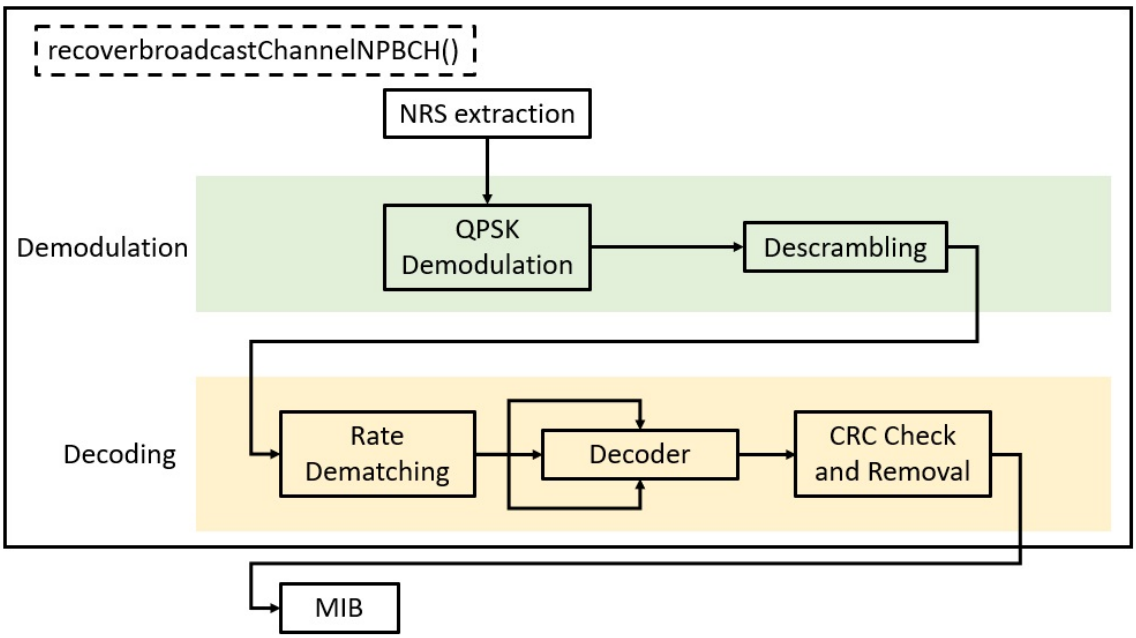


Figure 4.20: Block diagram of the MIB extraction procedure.

Function	recoverbroadcastChannelNPBCH
Input	npbch - recovered NPBCH subframe
	Ncell - Cell ID
Output	msbsfn - MSb of the hyper frame number
	hyperframen - LSb of the hyper frame number
	sib1schedsize - SIB1-NB scheduling data
	systemvaluetag - system value tag
	classbaring - class barring
	opmode - operation mode
	reserved - reserved bits

Table 4.14: recoverbroadcastChannelNPBCH function description.

4.6.3 NPDCCH

The procedures for demodulation and decoding are applied to the subframes containing the NPDCCH. The DCI formats are identified and the respective variables are extracted and delivered.

The identification of the DCI format is first made by identifying the size of the transmitted message. If the size is 23 bits then the format can either be N0 or N1 and its identified through its flag. If the size is 15 bits then it's a format N2 message. This assertion is made before the QPSK demodulation process to assure the descrambling process is made with the correct message length and sequence to make the message readable.

As shown in above sections some of the collected information in the DCI message is then used to identify the subframes containing the NPDSCH as well as the TBS for the NPDSCH.

An overview of the procedure is shown in figure 4.21, and the developed function is described in table 4.21.

Function	recoverControlChannel
Input	channel - NPDCCH subframe
	Ncell - Cell ID
	Nrnti - radio network temporary identifier
	nf - frame number
	ns - slot number
Output	recovered - DCI message

Table 4.15: recoverControlChannel function description.

4.6.4 NPDSCH

The subframes containing the NPDSCH are demodulated and decoded. The received bits may contain user-data or the SIB which are both delivered to the user.

The Shared Channel recovery depends on the scheduling information delivered by the NPDCCH by format N1, or the MIB for extracting the SIB1-N.

The process is shown in figure 4.22 and the developed function is described in table 4.16.

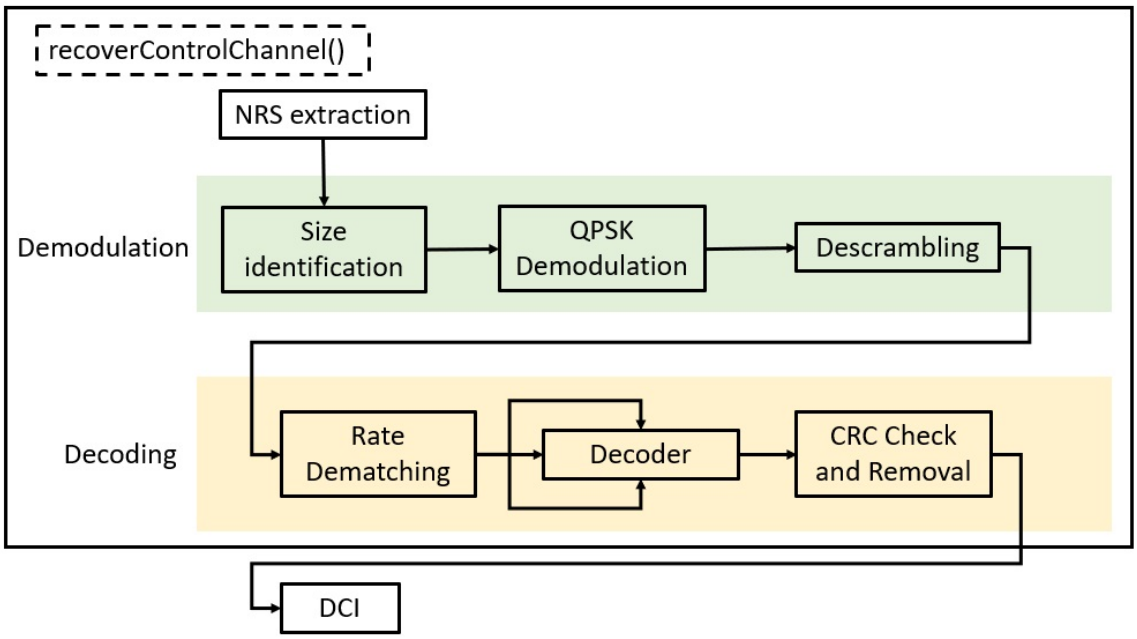


Figure 4.21: Block diagram of the DCI extraction procedure.

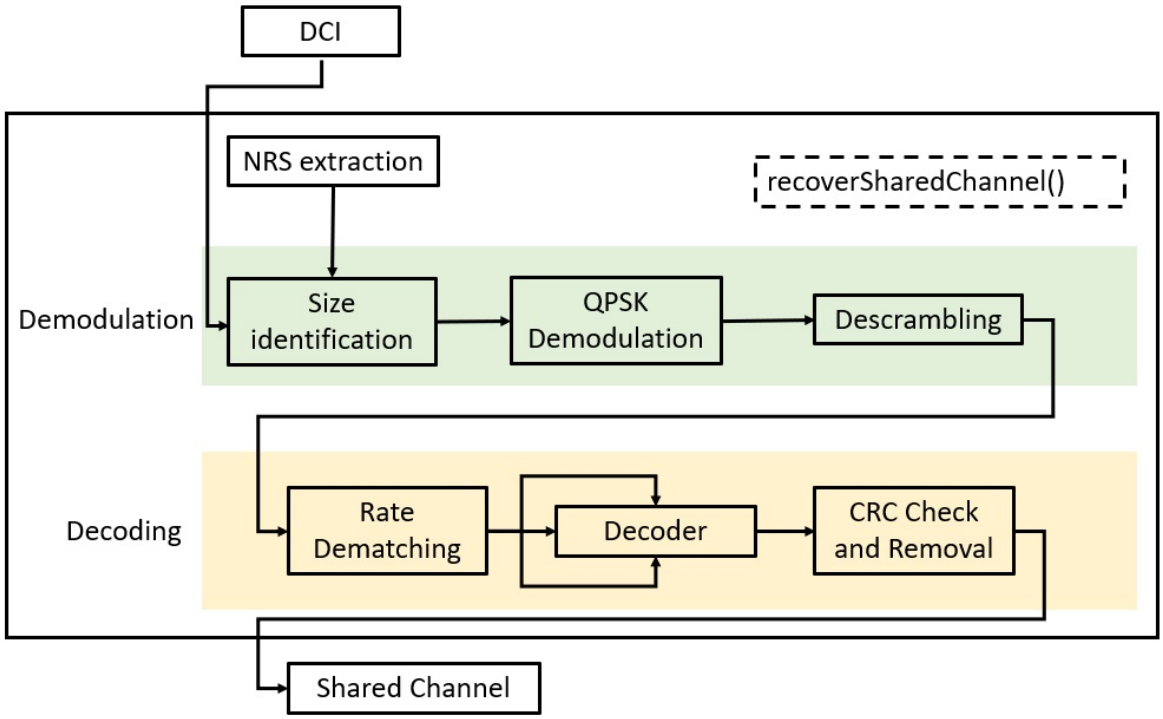


Figure 4.22: Block diagram of the Shared Channel extraction procedure.

Function	recoverSharedChannel
Input	channel - NPDCCH subframe
	Ncell - Cell ID
	Nrnti - radio network temporary identifier
	nf - frame number
	ns - slot number
	MNPDSCHrep - repetition number
	flagBCCH - flag for indicating if SIB is being carried
	tbs - Transport Block Size
Output	recovered - recovered data

Table 4.16: recoverSharedChannel function description.

The receiver is able to correctly recover the transmitted data following the standard's procedures. The recovered data from the MIB and DCI is used to recover the NPDSCH.

This chapter presented the procedures implemented for the recovery of the transmitted data. The following presents the integration of the behavioral model for a co-simulation environment.

Chapter 5

RF Front End Integration for Co-simulation

This chapter describes the SDR concept and the implemented co-simulation environment. SDR concept history and designs are presented with some considerations. The Radio Frequency (RF) front end choice is also presented.

5.1 Introduction

To provide a closer to reality implementation of NB-IoT a co-simulation environment was designed.

Implementing a co-simulation design allows to transmit/capture samples using actual hardware, bugs on the design can be detected and it gives a degree of comparison with the simulation-only environment.

The behavorial implementation on Matlab was used to both generate and recover NB-IoT baseband samples. The system was designed to operate using the generating function with one USRP board and the receiver with another.

The co-simulation is possible thanks to the SDR concept, which provides a background for developing flexible transceivers. The possibility of reducing the baseband samples generation to the digital domain, removes the real-time aspect, thus providing a method of testing and analysing. The integration with hardware allows for verification of the real signals, now respecting real-time conditions.

In the following sections, the SDR concept will be explained as well as the integration of the behavioral model with the SDR development kit.

5.2 Software Defined Radio

SDR defines a system for radio communications on which traditionally hardware implemented components are now implemented through software, such as mixers, filters, modulators, and others. It was introduced in the early 1990s by Joseph Mitola intenting to bring as much of possible of signal processing to the digital domain. With the SDR concept radio communication systems evolved towards more flexible implementations improving interoperabilty between devices and access points.

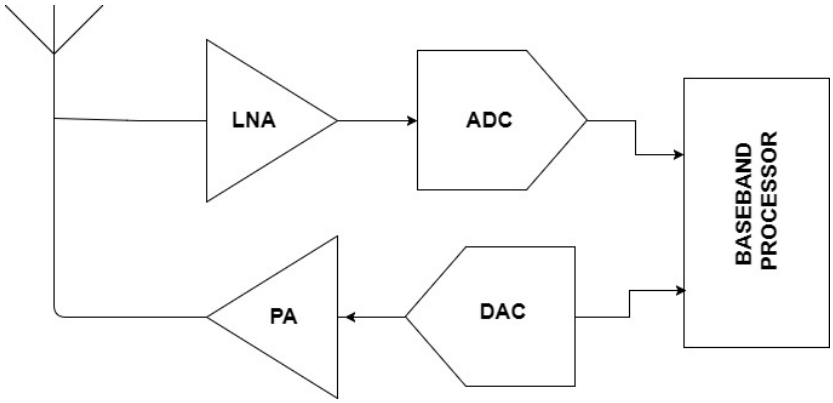


Figure 5.1: Ideal SDR system block diagram

As shown in figure 5.1 in an SDR system the analog domain is reduced to the strictly necessary such as the Low Noise Amplifier (LNA) for the receiver and the Power Amplifier (PA) for the transmitter. The signal conversion is made on either the Analog to Digital Converter (ADC) or Digital to Analog Converter (DAC), for the transmitter or receiver.

The baseband processing, the Physical Layer implementation, runs on digital domain based on Field Programmable Gate Array (FPGA)s, Digital Signal Processor (DSP)s or General Purpose Processor (GPP)s.

The SDR structure allows it to be seen as a mean to build multi-mode and multi-band equipment easily upgraded by software modification. It's a system on which functionality parameters such as frequency, power, modulation, radio technology are easily exchangeable or inter-combined by software. [Mit95]

The ideal SDR implementation, despite all its advantages, still poses many challenges for mainstream implementation. Nevertheless, the SDR concept provides a flexible architecture for radio design which can be easily changed and quickly upgraded. This enables new radio designs to support multi standard implementations and not only one-purpose solutions. In [Mon10] five factors are presented for justifying the predicted growth on SDR use.

1. **Multifunctionality** - The SDR concept brings the possibility of reconfiguring equipments to support different standards thus having multi purposes whereas analog radios would have one specific function. To change, update or add radio functions in SDR can be made by downloading the function whilst in analog radios new functions or standards are dependent on the initial function.

2. **Global Mobility** - Reconfigurable systems are able of changing and adapting, which enables the possibility of moving one system from one place to another and reconfigure it to adapt to new circumstances.

3. **Compressing capacity and energy efficiency**- SDR systems are able to provide more compact solutions which consequently lead to more energy efficient systems.

4. **Ease of construction** - SDR systems reduce the number of RF components on the architecture. This leads to a reduction of concerns regarding noise inserting components, thus reducing the complexity while designing the system.

5. **Constant update** - The growing number of updates and new coming standards leads to a continuous need of change and update. With SDR these changes are made through a download instead of hardware changes which disregards hardware limitations making updates easier and more affordable.

5.2.1 Topologies

The front end performs the Digital to Analog conversion on the transmitter side and the Analog to Digital conversion on the receiver side.

Ideally, on the receiver end, to fully digitize the received signal the ADC's sampling frequency and bandwidth would be infinite, which is not possible. To reduce these impossibilities other architectures are implemented.

The receiver side architectures vary by positioning the ADC in different positions and in the transmitter by varying the DAC's position. In this section the different receiver topologies are presented.

5.2.1.1 Baseband Digitalization

In the baseband digitalization architecture the ADC is placed in the baseband stage, which stands for SDR as a super-heterodyne receiver for analog radio.

It is the most commonly used architecture in radio receivers. The signal goes through two down conversion stages before reaching baseband frequency.

This architecture brings the advantage of minimizing the ADC requirements and also using narrowband components which present low power comsuption, however, by doing so it also narrows down the receiver's ability of broadband thus limiting the SDR to a single channel.

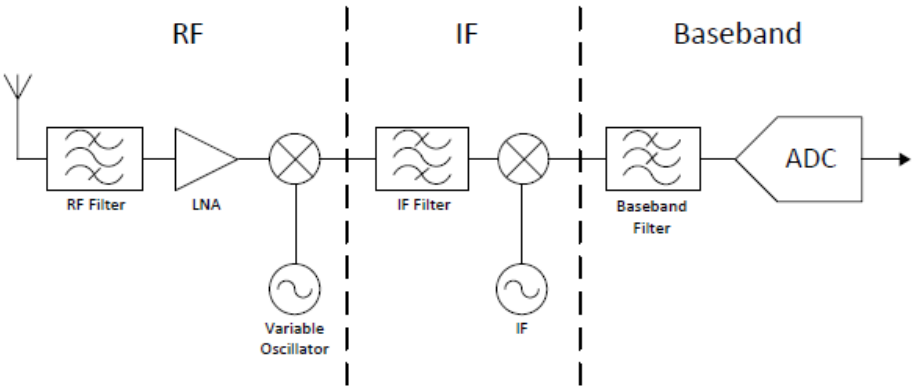


Figure 5.2: Baseband Digitalization block diagram [dSC10]

5.2.1.2 IF Digitalization

With this method the baseband stage is passed to the digital domain, placing the ADC in the Intermediate Frequency (IF) stage. The architecture reduces the number of components used and enables a wider bandwidth when compared to the previous architecture.

A down conversion to IF is performed, choosing the IF according to the ADC's sampling frequency. Even though power consumption rises, it allows for a bigger flexibility on the SDR enabling multi standard implementation.

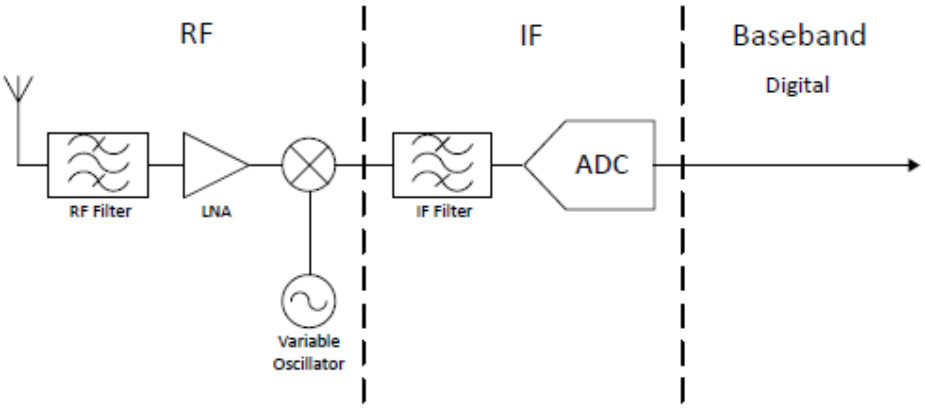


Figure 5.3: IF Digitalization block diagram [dSC10]

5.2.1.3 RF Digitalization

RF Digitalization is an approximation to the ideal SDR architecture.

In this architecture the ADC is moved to the RF stage which inherently brings issues regarding the ADC requirements. In order to digitize samples in the RF stage the ADC must comply with several specifications including high sampling rates and a large dynamic range. Nevertheless, this architecture is able to digitize the full spectrum and operate in virtually any standard thus realizing the SDR concept in full.

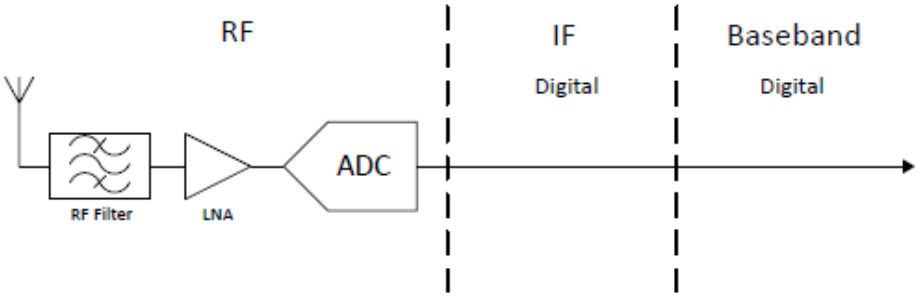


Figure 5.4: RF Digitalization block diagram [dSC10]

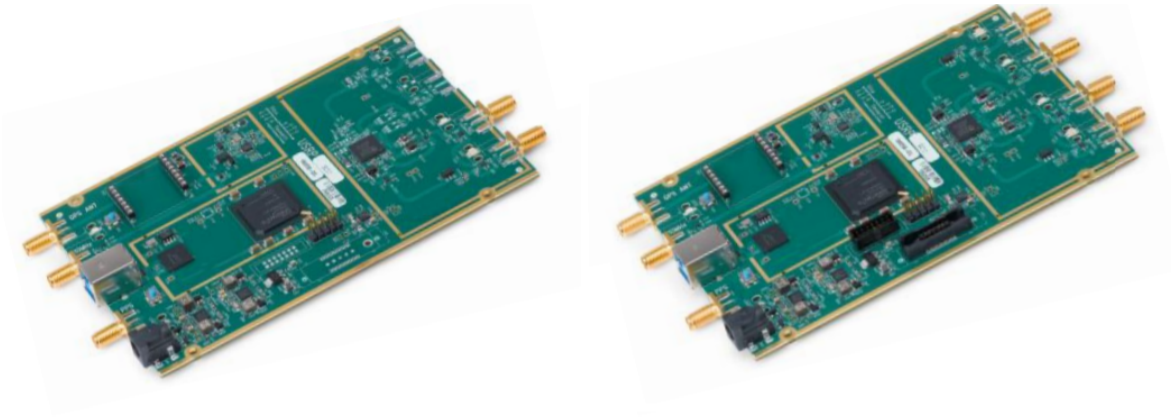
5.2.2 Development Kits

For SDR to be implemented a radio front end equipped with the elements present in figure 5.1 is needed. Commercial front ends are based on FPGA and present various integrations methods with DSPs and GPPs.

Front ends vary on several parameters such as frequency range, instantaneous bandwidth, Multiple Input Multiple Output (MIMO) support, ADC/DAC conversion time, price, etc...

5.2.2.1 USRP

The USRP is an SDR platform for prototyping and testing radio signals. The USRP is a product manufactured by Ettus Research^{TM}, a company of National Instruments (NI). The USRP boards used were the USRP B200 and B210. [Res17b]



(a) USRP B200 board

(b) USRP B210 board

Figure 5.5: USRP Boards used

Both boards operate on a frequency range from 70MHz-6GHz. The boards are based on a Spartan6 FPGA and are capable of transmitting various signals such as FM, cellular, Wi-Fi and more. The main difference between the USRP B200 and USRP B210 is that USRP B200 supports only one Tx and one Rx channel while USRP B210 supports two of each, supporting MIMO.

The main advantage of the USRP is the USRP Hardware Driver (UHD) [Res17a], which is an application that supports development and portability for the use of the USRP with different SDR platforms.

5.3 Integration with USRP for Co-simulation

The chosen board for the co-simulation was the USRP. It presents an easy integration with *.m* files, which was a crucial demand for this dissertation, it has a wide range of frequencies which allows for testing in the E-UTRA spectrum. The USRP B210 also supports MIMO which is useful for further implementations. In order to input the baseband samples to the USRP the UHD was installed. This installation was made through Matlab's communication toolbox.

The integration with Matlab allows to build a radio object. The radio object allows to define properties in the USRP such as the sampling rate, the center frequency, gain and others.

To perform the signal transmission and reception using an SDR platform two USRP boards were needed as well as two laptops. The setup used is shown in figure 5.6:

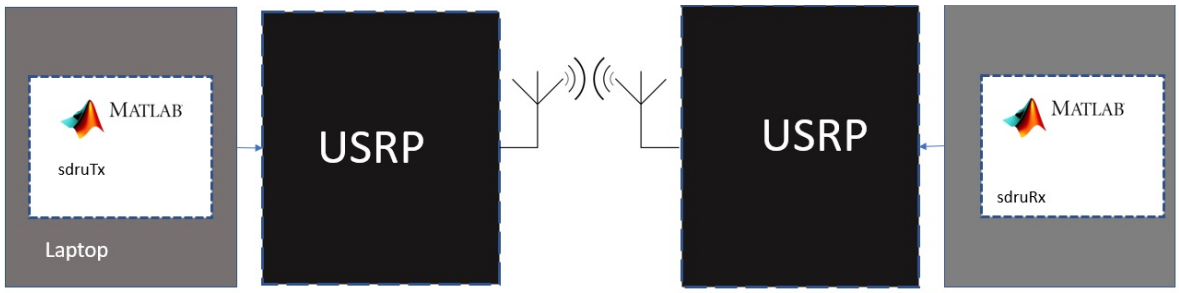


Figure 5.6: Block diagram of the setup used

5.3.1 Transmitter

To transmit the NB-IoT generated signals through the USRP a new script was built, *sdruTX.m* [Mat17e]. This script calls the previous function generateDLsignal.m to import the baseband samples. The SDR radio object is then created with conditions shown in figure 5.7:

```
                ChannelMapping: 1
       CenterFrequencySource: 'Property'
             CenterFrequency: 900000000
       ActualCenterFrequency: 900000000
LocalOscillatorOffsetSource: 'Property'
      LocalOscillatorOffset: 0
ActualLocalOscillatorOffset: 0
                 GainSource: 'Property'
                       Gain: 25
                 ActualGain: 25
                ClockSource: 'Internal'
           MasterClockRate: 7680000
     ActualMasterClockRate: 7.6800e+06
  InterpolationFactorSource: 'Property'
        InterpolationFactor: 4
  ActualInterpolationFactor: 4
          TransportDataType: 'int16'
        UnderrunOutputPort: 1
           EnableBurstMode: 0
```

Figure 5.7: USRP transmitter side set of properties

The center frequency chosen was 900MHz which is a GSM band carrier. This choice was made in order to fully design the standalone deployment mode. The interpolation factor is 4 in order to set the sampling rate of the transmitted to 1.92MHz as desired. The interpolation factor decimates the Master Clock Rate by a factor of 4.

The decision of a 25 dB gain was made by considering Matlab's LTE transmitter, since NB-IoT operates in the licensed spectrum then the transmitted power restrictions are similar to LTE, although for the scope of this dissertation power control wasn't taken in consideration.

The transport data type is set automatically by the UHD and stands for the format which the baseband samples are carried from the Matlab environment to the USRP. The components are divided in I and Q components in the form of 16 bits integers before being upconverted.

Having the baseband samples ready and the radio object defined the transmission can begin. The procedure is made by sweeping the baseband samples slot by slot while performing the Matlab's step function between the radio object and the slot.

The procedure is defined to last for 3 minutes repeating the same radio frame on loop. Prior to the transmission a spectrum analyzer object is also created in order to analyze the spectrum of the transmitted signal. An example is shown in figure 5.8 and is further analysed in the results chapter.

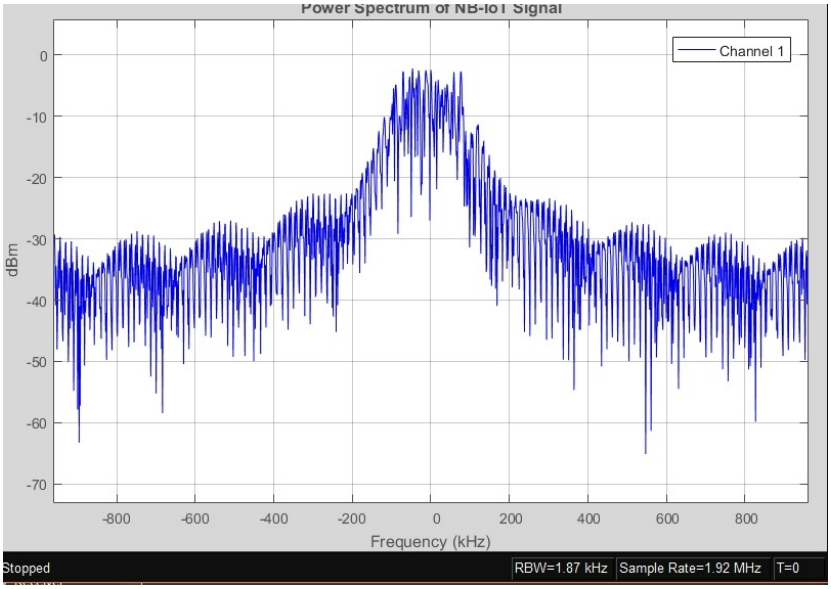


Figure 5.8: Example of a transmitted NB-IoT signal spectrum

The implementation of the `sdruTx` function can be described in the block diagram of figure 5.9

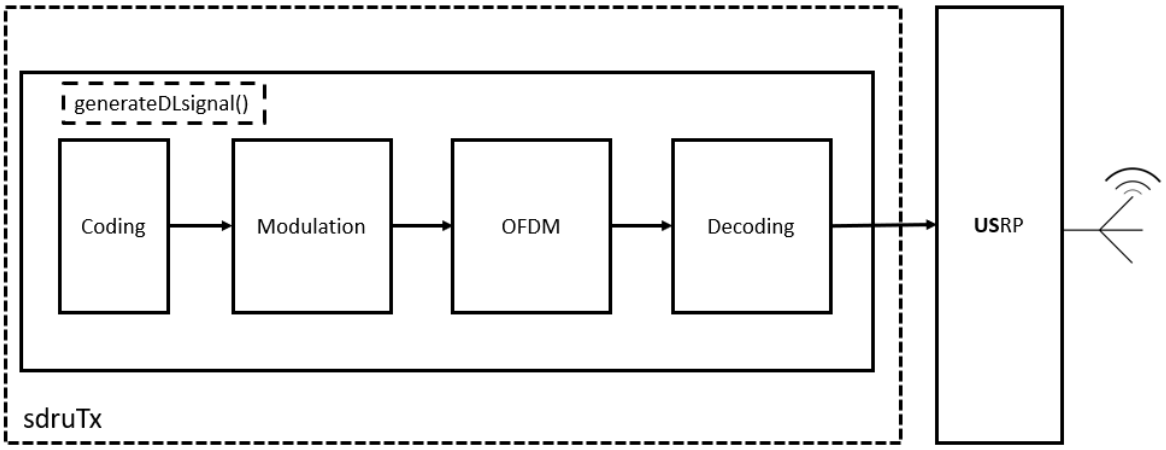


Figure 5.9: Block diagram of co-simulation transmitter

5.3.2 Receiver

The receiver is implemented in *sdruRx.m* set to capture a number of samples fulfilling one radio frame. To receive the samples another radio object needs to be defined, setting it to receive samples.

The radio object is defined with the properties shown in figure 5.10:

The center frequency had to be 900MHz since it's the frequency set for transmission. The decision of a 30 dB gain was also made by considering Matlab's LTE receiver. The transport data type is set automatically by the UHD and stands for the format which the baseband samples are carried from the USRP board to the Matlab environment. Upon reception the samples are downsconverted to baseband samples and divided in I and Q components in the form of 16 bits integers.

After defining the radio object a capture window buffer is created the size of a radio frame. The buffer is shaped in a matrix of 960 rows of 20 columns, meaning it collects each slot of the radio frame in a column. [Mat17d]

Since the radio object is defined to receive, by applying Matlab's step function to the object the received samples are then stored in the created buffer.

Similar to what was done in the transmitter a spectrum analyzer object is also created to visualize the received signal's spectrum. An example is shown in figure 5.11, and is further analyzed in the results chapter.

```
comm.SDRuReceiver with properties:

                    Platform: 'B210'
                  SerialNum: '30FDE65'
             ChannelMapping: 1
       CenterFrequencySource: 'Property'
             CenterFrequency: 900000000
       ActualCenterFrequency: 900000000
 LocalOscillatorOffsetSource: 'Property'
       LocalOscillatorOffset: 0
 ActualLocalOscillatorOffset: 0
                 GainSource: 'Property'
                       Gain: 30
                 ActualGain: 30
                ClockSource: 'Internal'
           MasterClockRate: 7680000
      ActualMasterClockRate: 7.6800e+06
     DecimationFactorSource: 'Property'
           DecimationFactor: 4
     ActualDecimationFactor: 4
          TransportDataType: 'int16'
          OverrunOutputPort: 1
                 SampleRate: 1
             OutputDataType: 'double'
                FrameLength: 960
              EnableBurstMode: 1
            NumFramesInBurst: 20
```

Figure 5.10: USRP receiver side set of properties

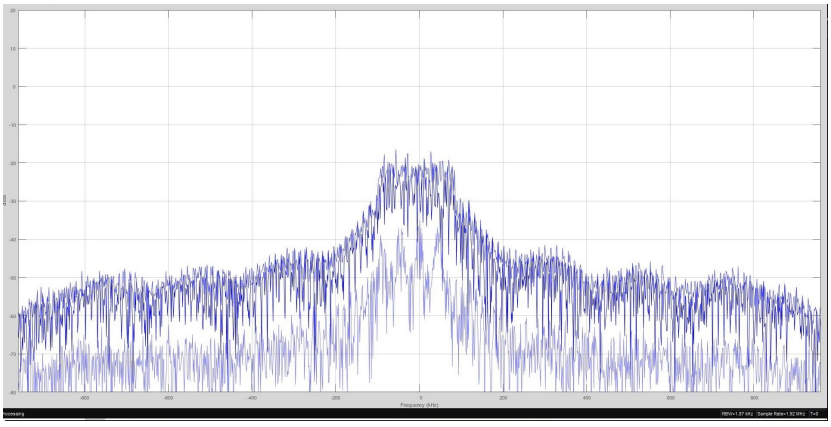


Figure 5.11: Example of a received NB-IoT signal spectrum

In `sdruRx` the collected samples are then reshaped into a line and set as input to the frame synchronization procedures and afterwards to the data recovery function, *recoverDLSignal.m* as show in figure 5.12

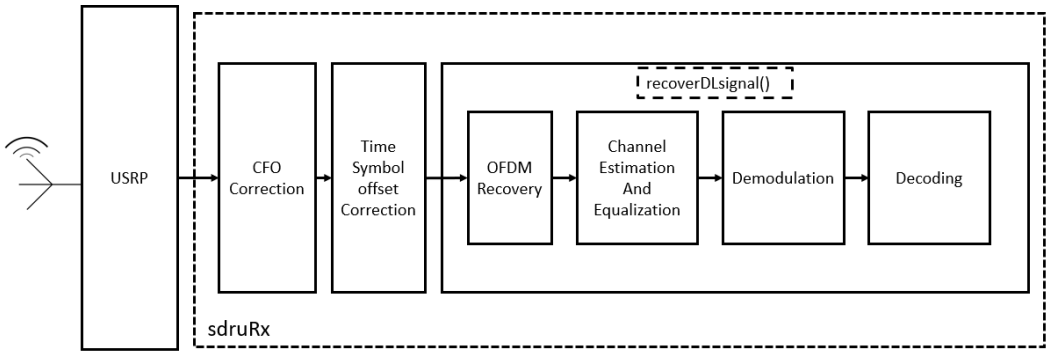


Figure 5.12: Block diagram of the co-simulation receiver

The final implementation can be described by figure 5.13. The baseband samples are generated in the Matlab, are then upconverted digitally by USRP and converted to Analog data and transmitted by the USRP RF daughter board.

For the receiver, the RF daughter converts the Analog data back to digital in their I and Q components, which digitally downconverted and then recovered by the Matlab behavioral model.

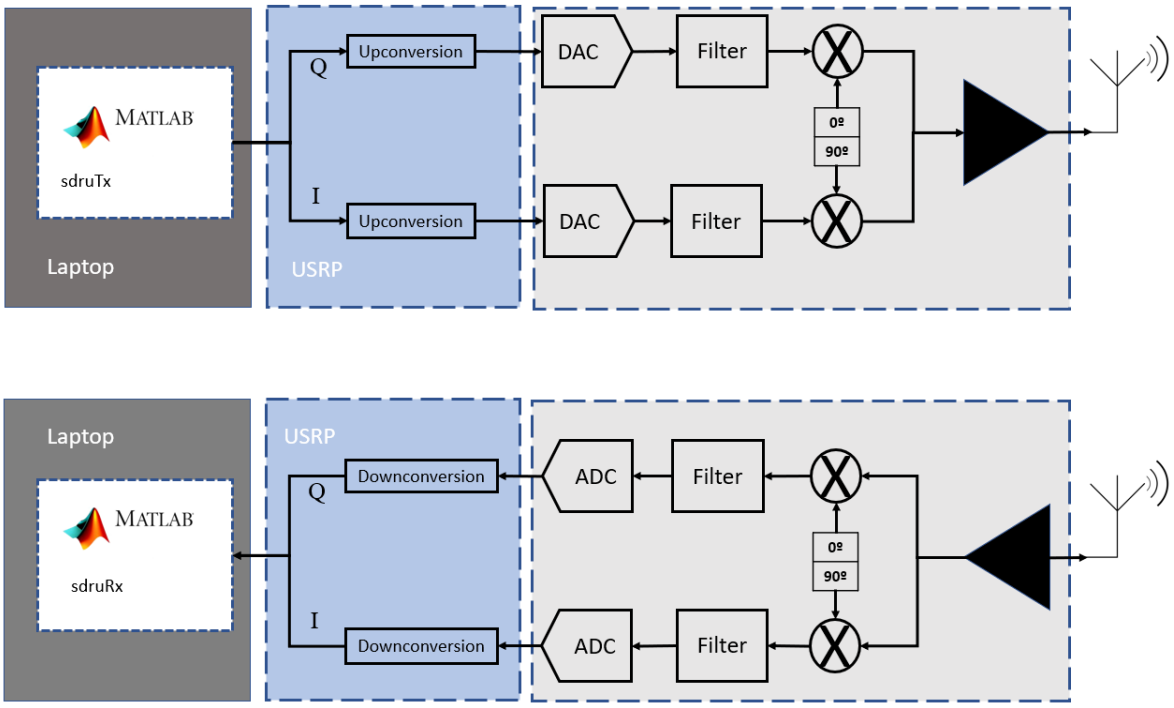


Figure 5.13: Block diagram of the SDR implementation for the USRP co-simulation design

The co-simulation was successfull, the transmission and reception is made correctly, the received data is equal to the transmitted data. The results for the applied setup are presented in the next chapter.

Chapter 6

Results

In this chapter the tests to which the design was subjected to are presented along with the obtained results and commentary on them. An evaluation of Z-F equalizer is presented by analysing constellation and eye diagrams of the received signal. Bit Error Rate (BER) performances are also taken to assessment.

6.1 Introduction

In order to further assess the system's behavior and response to real-life situations regarding noise and multi path fading channel models needed to be applied in order to predict the situations and mitigate their effects. To evaluate the system BER is calculated to provide the answer to "to what extent does the system work?".

The Z-F equalizer plays a major part on the correct recovery of the samples hence the tests are focused on its behavior and performance in order to assure its purpose.

Constellation and eye diagrams provide information regarding the samples quality, in the sense that the presence of noise causes scattering in the constellations and unclear eye diagrams.

The next sections present the channel models applied along with the results for each one considering the evaluation tools above. Further sections present the co-simulation with hardware in the result results.

6.2 Simulation Results

The developed environment for simulating the transmission is described in figure 6.1.

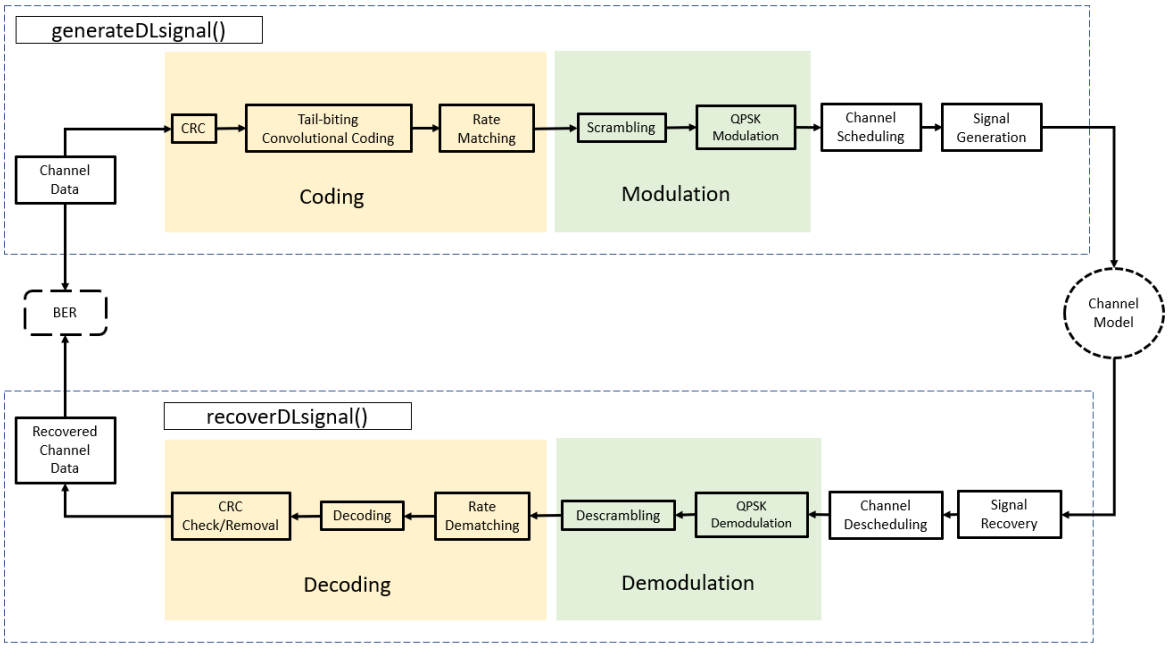


Figure 6.1: Block diagram of the simulation environment

The simulation script calls the generating and recovering functions for the NB-IoT downlink samples, applies a channel model (Rayleigh and AWGN, or just AWGN).

The simulation requires only one Matlab session which allows for BER evaluation. Constellation and eye diagrams are also presented and serve as a mean to assess the transmissions' quality and feasibility.

6.2.1 Channel Models

In order to simulate real life conditions a channel simulation must be made. Channel provides are a tool to predict the system's behavior when the signals experiment different scenarios.

6.2.1.1 AWGN

The most common cause of error in communication systems is the signal's addition to random values caused by thermal noise, for example. The simulation for this situation can be made using an AWGN channel, for which the received signal is given by (6.1).

$$y(t) = x(t) + n(t) \tag{6.1}$$

Where y is the received signal, x the transmitted signal and n the additive noise. The model was applied using Matlab's `awgn` function.

6.2.1.2 Multipath Fading Model

The signal traveling from transmitter to receiver may be facing multiple reflective paths leading to multipath fading. This causes the received signal to present changes in amplitude, angle and phase. These effects are classified as slow fading/fast fading and frequency-selective/flat fading channels.

In order to minimize the effects models can be used to help predict them and therefore work towards correcting them. In this dissertation the model chosen was the Rayleigh Flat Fading Channel [Wav17a].

Under this model, the received signal is given by equation (6.2).

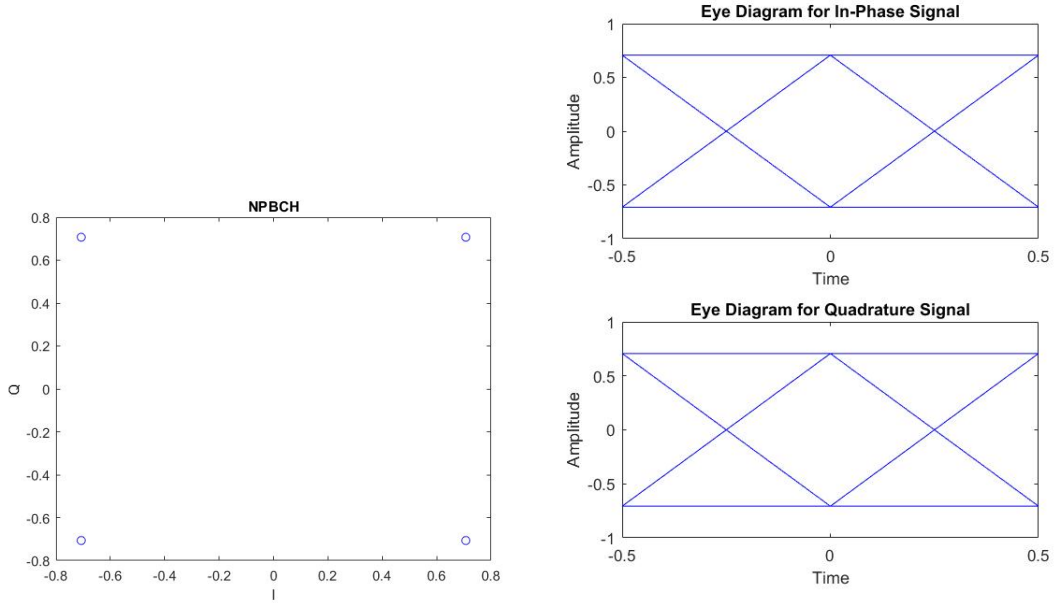
$$y(t) = hx(t) + n(t) \tag{6.2}$$

Where y is the received signal, x the transmitted signal, h the Rayleigh fading factor and n the AWGN noise. This was applied using matlab's `rayleighchan` and `awgn` functions.

6.2.2 Transmitter

The presented results are for a radio frame containing the NPBCH, the NPDCCH format N1 message scheduling the NPDSCH for a 72 bits Transport Block. The radio frame also contains both the NPSS and NSSS.

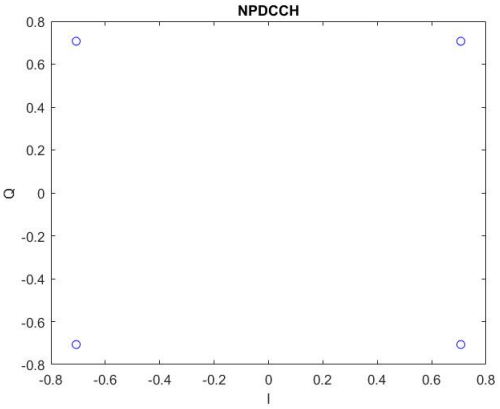
The generator script outputs the constellation and eye diagrams for the generated frame.

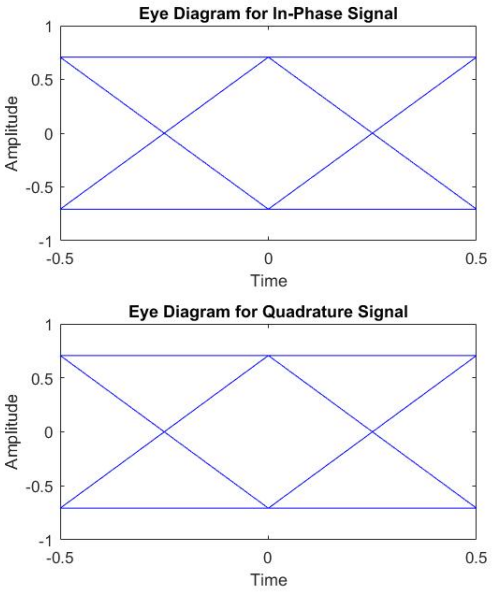


(a) Constellation diagram for the transmitted NPBCH

(b) Eye diagram for the transmitted NPBCH

Figure 6.2: Transmitted NPBCH

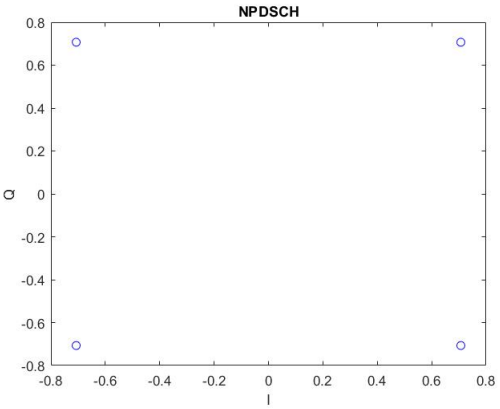


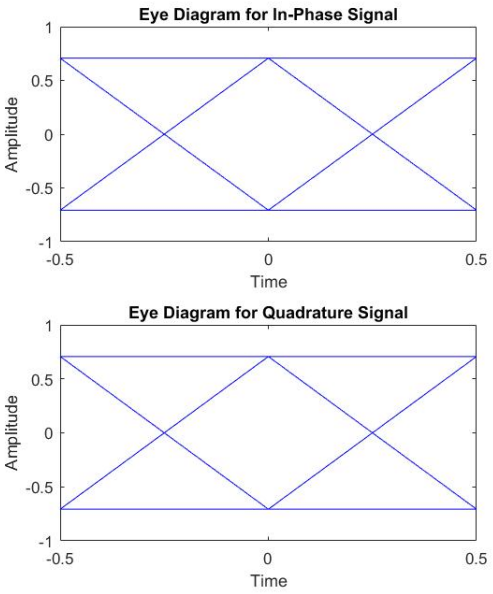


(a) Constellation diagram for the transmitted NPDSCH

(b) Eye diagram for the transmitted NPDCCH

Figure 6.3: Transmitted NPDCCH





(a) Constellation diagram for the transmitted NPDSCH

(b) Eye diagram for the transmitted NPDSCH

Figure 6.4: Transmitted NPDSCH

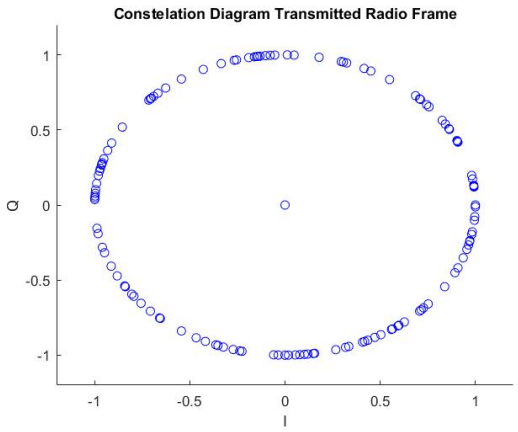


Figure 6.5: Constellation diagram for the transmitted radio frame

Figures 6.4, 6.3 and 6.2 show the constellation and eye diagrams of the generated samples for each physical channels.

As is observed in figure 6.5 the Zadoff-Chu sequences for the NPSS and NSSS are present. The QPSK symbols are not very noticeable since they are all overlapped.

The center point is expected and corresponds to the inserted zeros for unused resource elements.

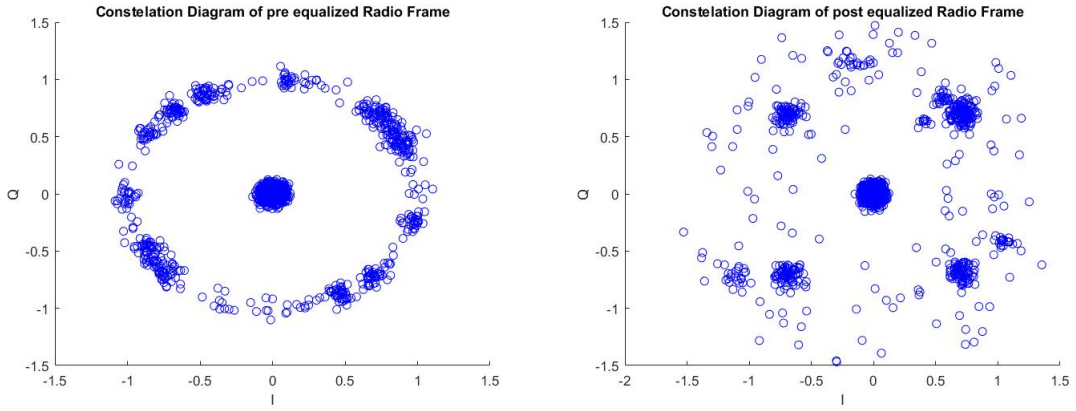
6.2.3 Receiver

The generated samples were subjected to test to verify the Z-F equalizer performance. A Rayleigh channel model was applied with conditions shown in table 6.1 and AWGN added noise with SNR=10dB.

Name		Value
`ts`	Sampling Time (μ s)	0.52
`fd`	Maximum Doppler shift (Hz)	70
`tau`	Path Delays (ns)	0 5 12 20 23 50 160 230 500
pdb	Path Gain (dB)	-1.0 -1.0 -1.0 0.0 0.0 0.0 -3.0 -5.0 -7.0

Table 6.1: Input values for generating a Rayleigh channel with Matlab function rayleighchan

After applying the channel model effects the received radio frame before and after equalization is presented in figure 6.6 as output of the recovery function.



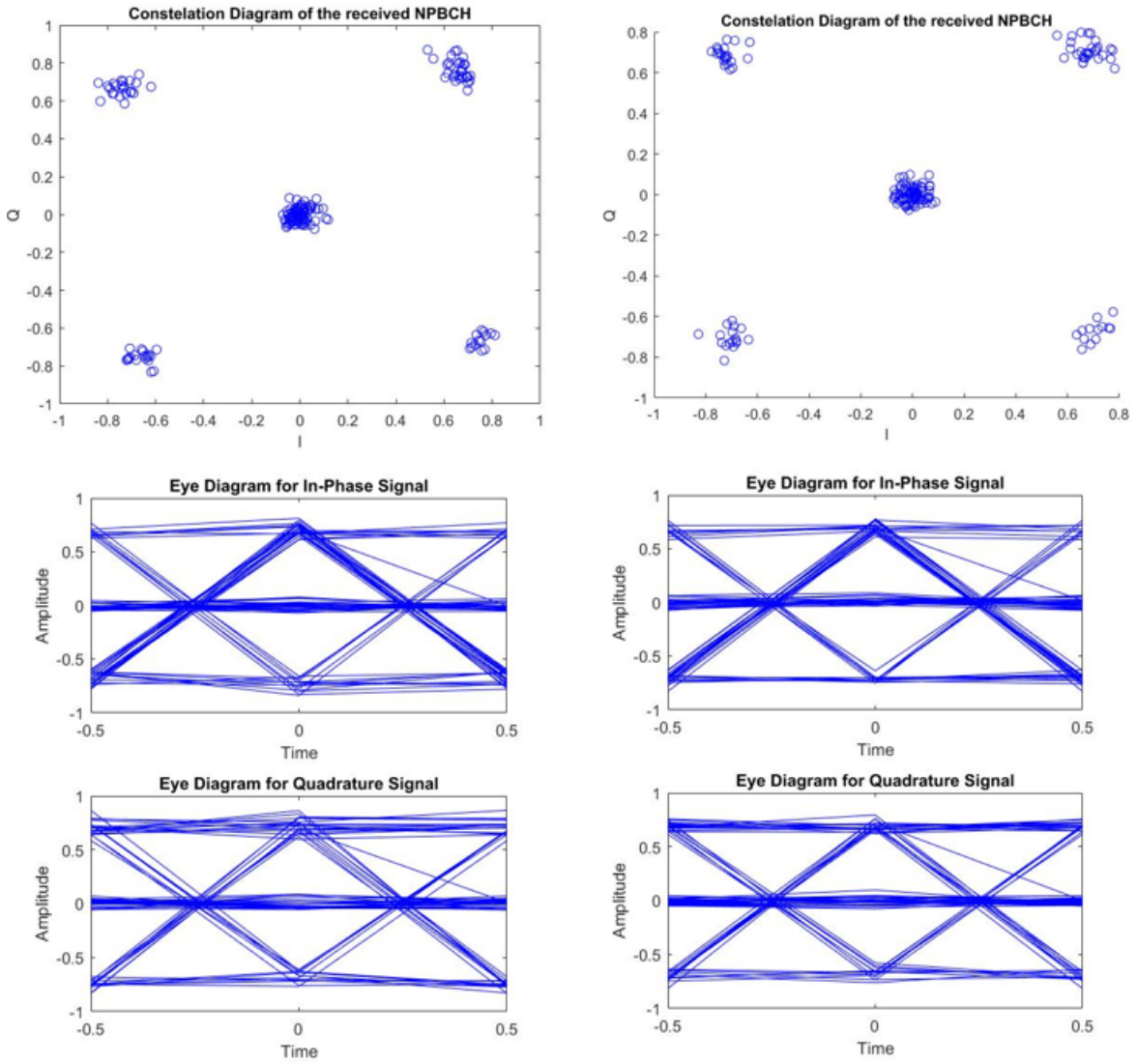
(a) Full Frame constellation diagram pre equalization

(b) Full Frame constellation diagram post equalization

Figure 6.6: Full received radio frame pre and post equalization

It's now noticeable the QPSK symbols in their reference regions. The Zadoff-Chu samples are now scattered which is expected since there is no NRS values present in the NPSS or NSSS subframes.

After the descheduling procedure the recovery function also outputs the constellation and eye diagrams for the received channel's subframe.

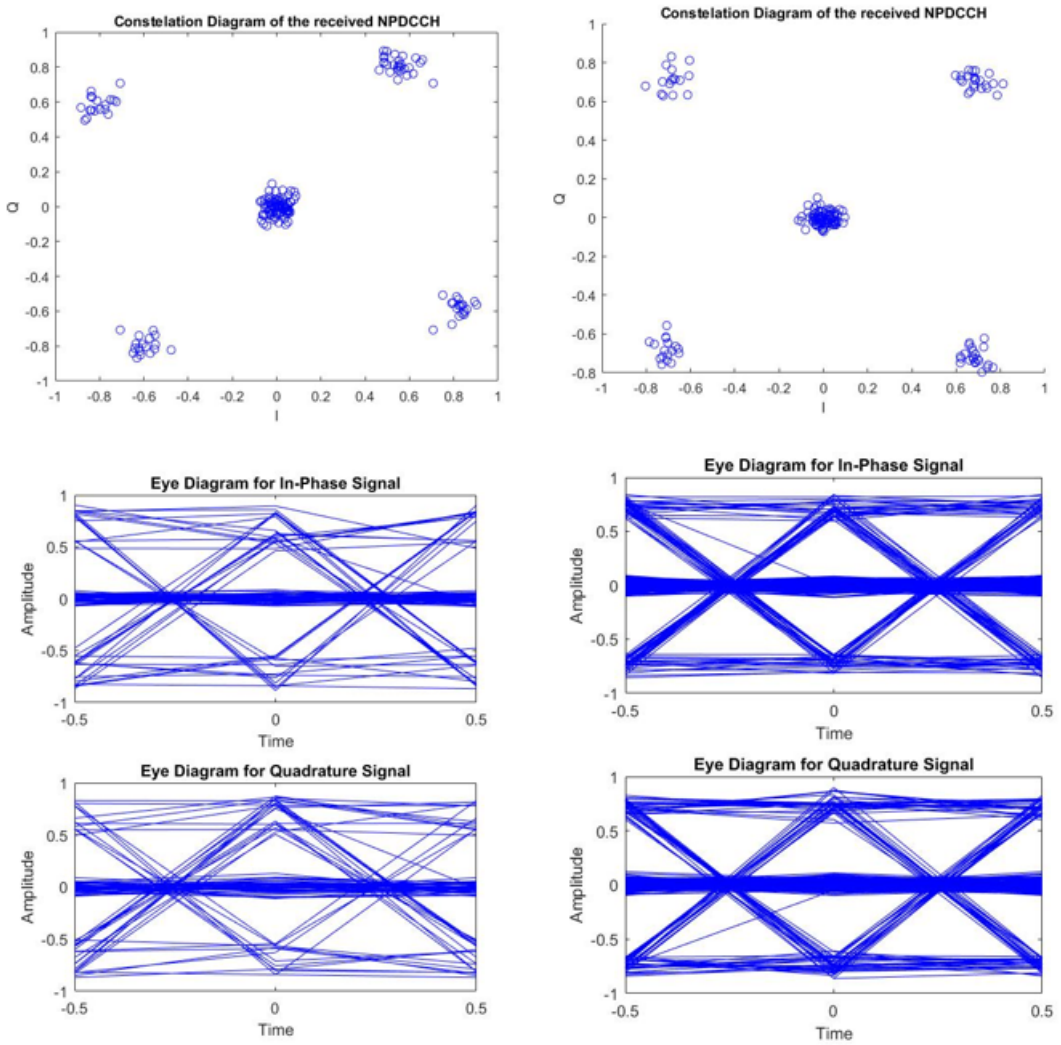


(a) Constellation and eye diagram of the received NPBCH without equalization

(b) Constellation and eye diagram of the received NPBCH with equalization

Figure 6.7: Received NPBCH subframe

The NPBCH is present in the first subframe, subframe 0. The performance for extracting the MIB in the NPBCH is clearly better with equalization. The received symbols are clearly QPSK, however, a slight frequency shift is noticeable in figure 6.7 a) and the eye diagram shows poorer quality when compared to b).



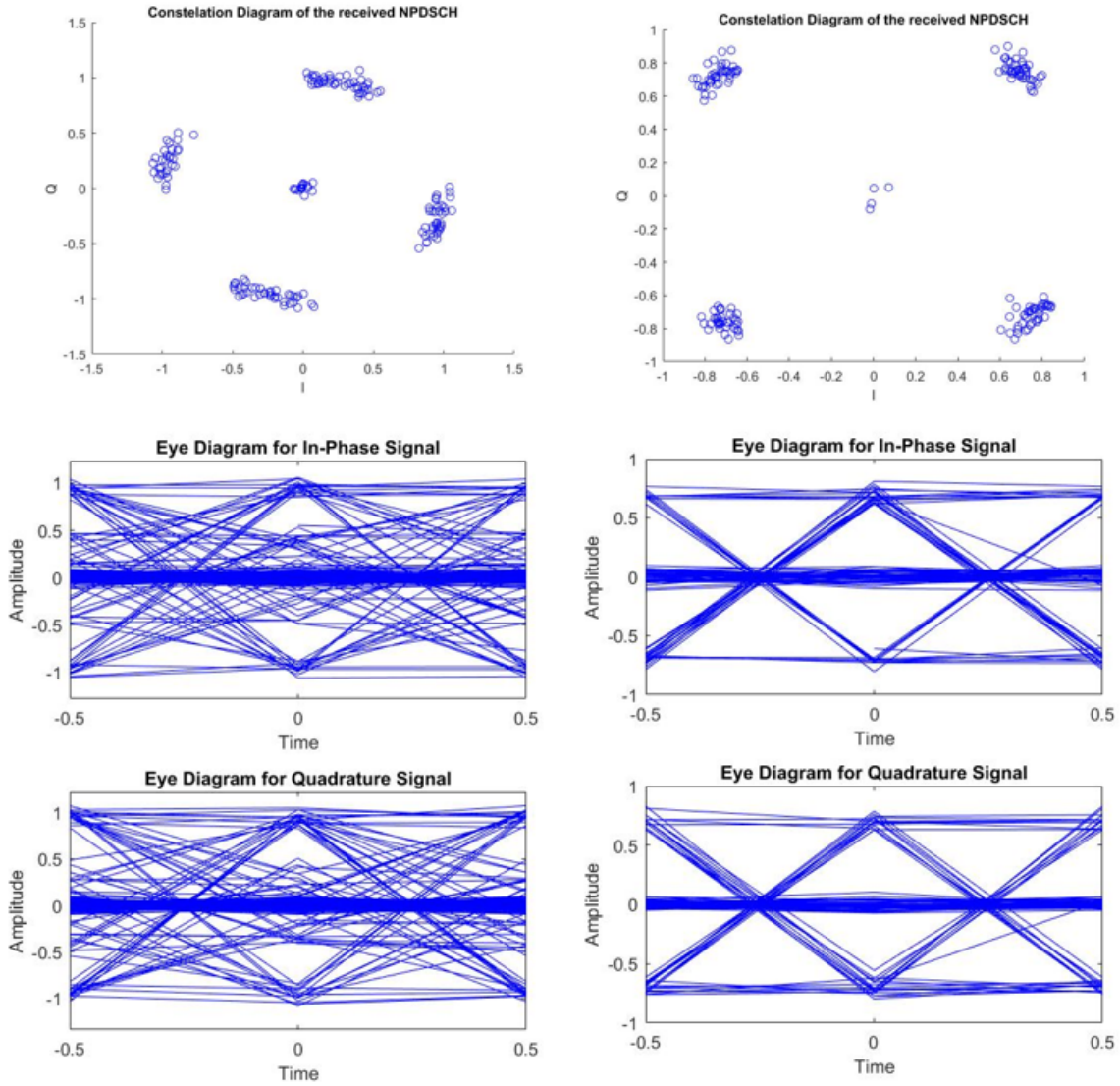
(a) Constellation and eye diagram of the received NPDCCH without equalization

(b) Constellation and eye diagram of the received NPDCCH with equalization

Figure 6.8: Received NPDCCH subframe

The NPDCCH is present in the second subframe, subframe 1. The performance for extracting the DCI format in the NPDCCH is also clearly better with equalization. The received symbols are clearly QPSK, however, a slight frequency shift is also noticeable in figure 6.7 a) and the eye diagram shows poorer quality when compared to b).

A wrong extraction of the DCI message severely compromises the extraction of the NPDSCH since the scheduling information for the NPDSCH is carried in the NPDCCH carrying the N1 format.



(a) Constellation and eye diagram of the received NPDSCH without equalization

(b) Constellation and eye diagram of the received NPDSCH with equalization

Figure 6.9: Received NPDSCH subframe

The NPDSCH is present in the seventh subframe, subframe 6. The performance for extracting the NPDSCH is seemingly impossible without equalization. The received symbols are shifted of the QPSK reference regions. The eye diagram in figure 6.9 a) clearly shows the poor quality on the transmission which obviously informs the impossibility of correctly receiving the data without the equalizer. The process for extraction may also have been jeopardized due to a false information given by the wrongly extracted DCI format. Observing figure 6.9 b) it's obvious the reception is performed correctly.

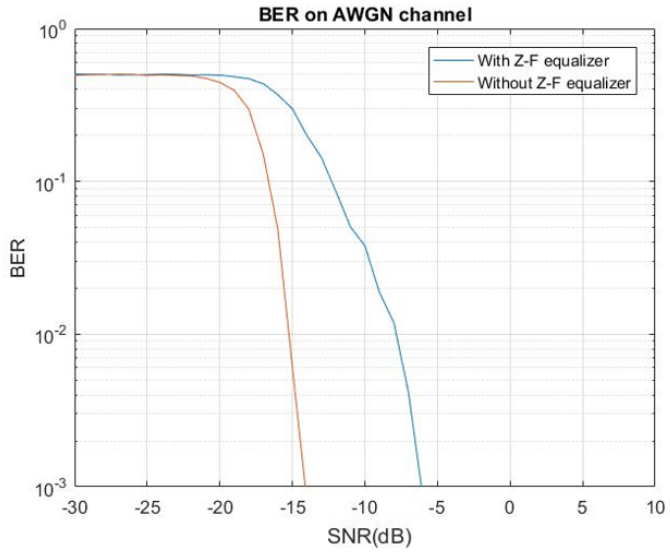
The received subframes presented in figures 6.7 b), 6.8 b) and 6.9 b) clearly present QPSK symbols in their reference regions and also open and clear eye diagrams which leads to conclude the equalizer's need and correct performance.

6.2.4 BER Performance

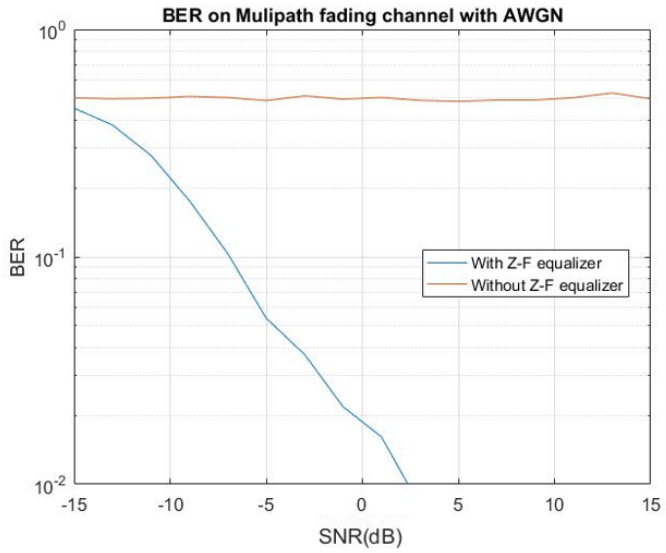
The BER determines the ratio of erroneous bits over the total of transmitted bits.

$$BER = \frac{Error\ Bits}{Number\ of\ transmitted\ bits} \tag{6.3}$$

In this subsection the BER is calculated for various Signal-to-Noise Ratio (SNR) values indicating the systems performance for different channel conditions.



(a) BER results for transmission of a radio frame on an AWGN channel with and without Z-F equalizer

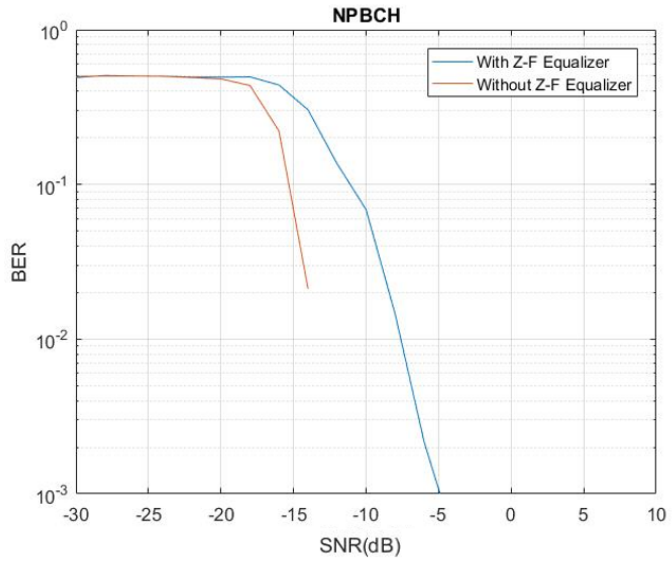


(b) BER results for transmission of a radio frame on a Multipath fading with AWGN channel with and without Z-F equalizer

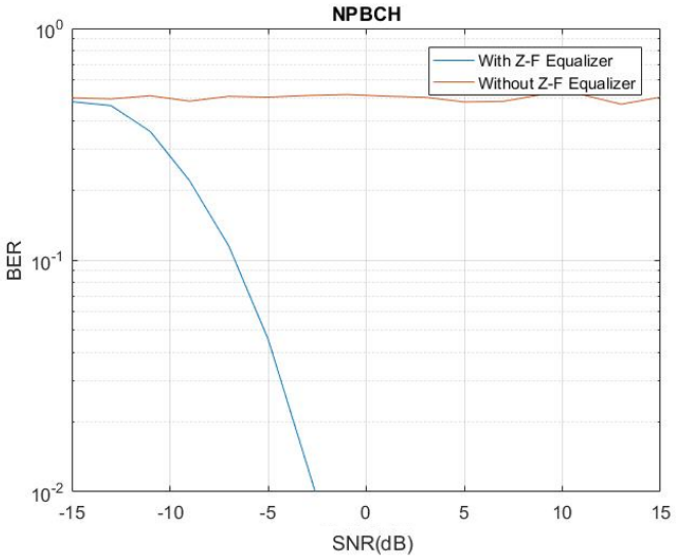
Figure 6.10: BER performances for the full radio frame

As expected the system presents a better performance when not using the Z-F equalizer for the AWGN channel in figure 6.10 a). This is because the Z-F channel estimation does not compensate for additive noise. When applying a Multipath fading channel with the parameters described in table 6.1 the reception is only possible when using the equalizer although severely jeopardized.

The same channels models were applied for evaluating the reception of each NB-IoT physical channel.

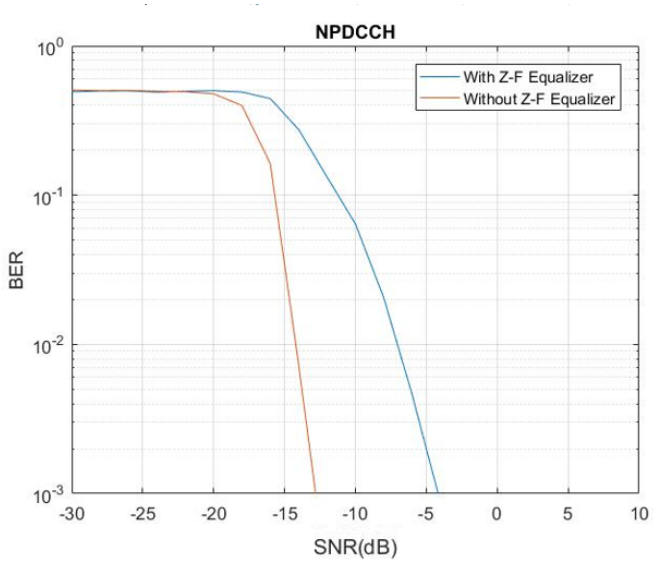


(a) BER results for transmission of the NPBCH on an AWGN channel with and without Z-F equalizer

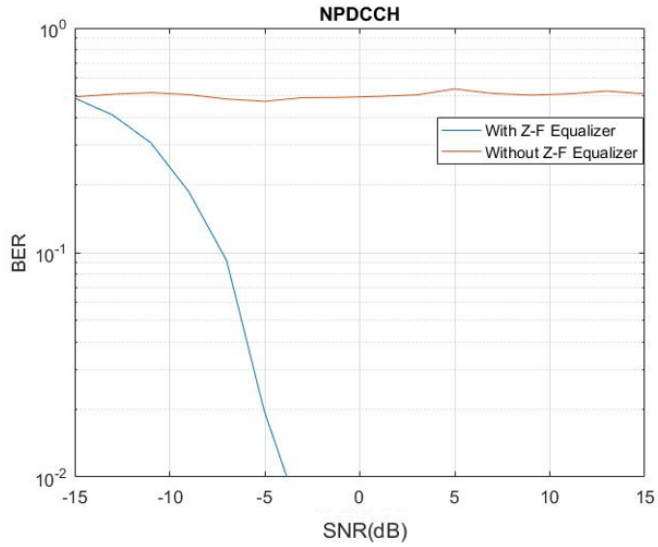


(b) BER results for transmission of the NPBCH on a Multipath fading with AWGN channel with and without Z-F equalizer

Figure 6.11: BER performances for the NPBCH

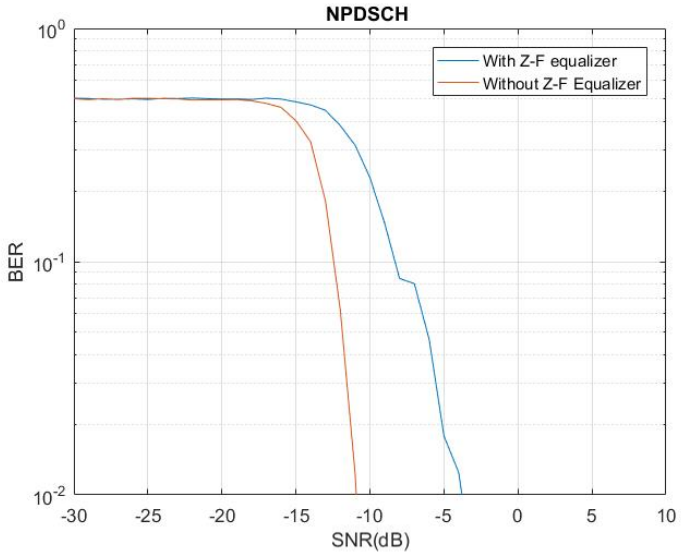


(a) BER results for transmission of the NPDCCH on an AWGN channel with and without Z-F equalizer

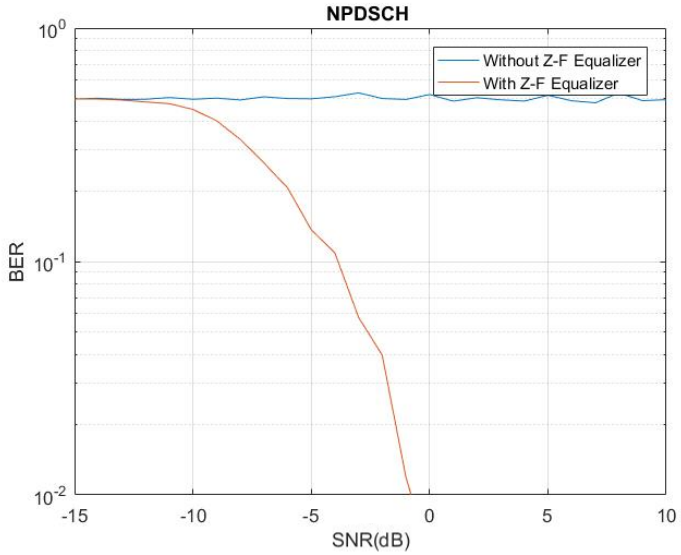


(b) BER results for transmission of the NPDCCH on a Multipath fading with AWGN channel with and without Z-F equalizer

Figure 6.12: BER performances for the NPDCCH



(a) BER results for transmission of the NPDSCH on an AWGN channel with and without Z-F equalizer



(b) BER results for transmission of the NPDSCH on a Multipath fading with AWGN channel with and without Z-F equalizer

Figure 6.13: BER performances for the NPDSCH

The results for each channel were as expected. The equalizer's performance is consistent. The NPBCH and NPDCCH present better results when the system is subjected to a multipath fading model when compared to the NPDSCH.

To assess the effect of repetitions on the transmission of the NPDSCH a fixed sequence of 72 bits was transmitted varying the number of repetitions each time. The repetition values used were 1, 4, 16, 32, 64 and 128. The simulation applied an AWGN channel to the transmitted channel.

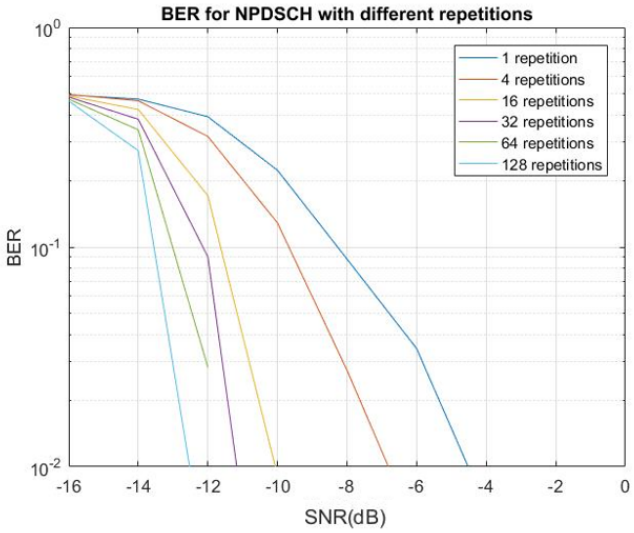


Figure 6.14: BER results for transmission of the NPDSCH for a number of repetitions of 1, 4, 16, 32, 64 and 128 on an AWGN channel

In figure 6.14 the results demonstrate that for each increase of 2^x repetitions for $x + 1$ theres a BER gain of approximately 2dB, altough for a higher number of repetitions the gain decreases to around 1dB difference.

6.3 USRP Co-simulation Results

The integration with the USRP was successful, the code is able to transmit and receive samples correctly delivering correct data to the user.

For the transmission using the USRP the obtained samples were correctly demodulated and decoded delivering the correct transmitted data to the user.

The setup used is demonstrated in 6.15

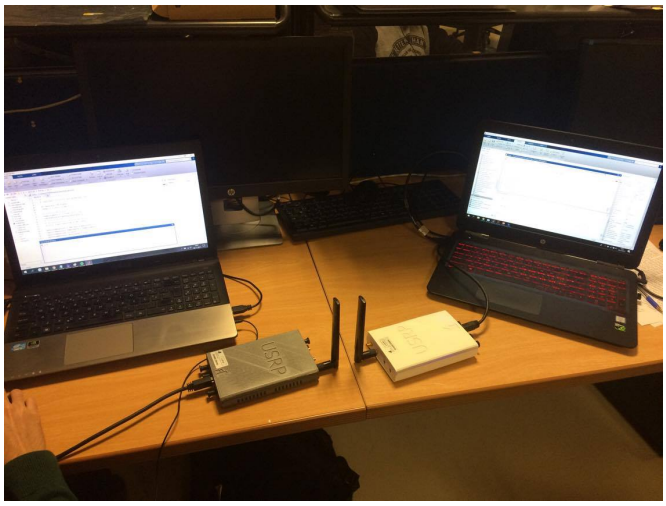


Figure 6.15: Setup used for co-simulation

The received samples were as follows:

Firstly, the frame synchronization procedures were applied. The frequency offset detected a 2336.636Hz offset and the Time symbol offset was of 4998 samples. The correction was performed successfully. The time symbol offset correct is shown in figure 6.16.

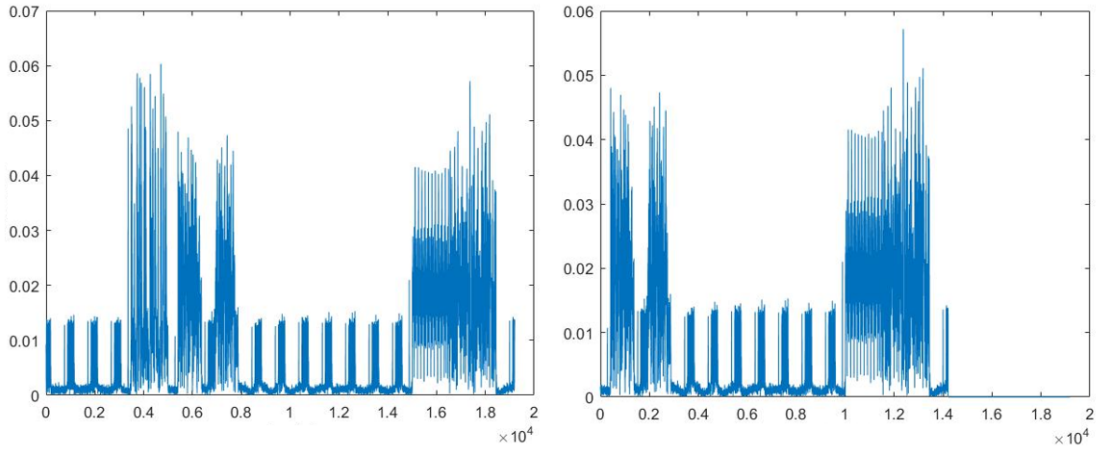
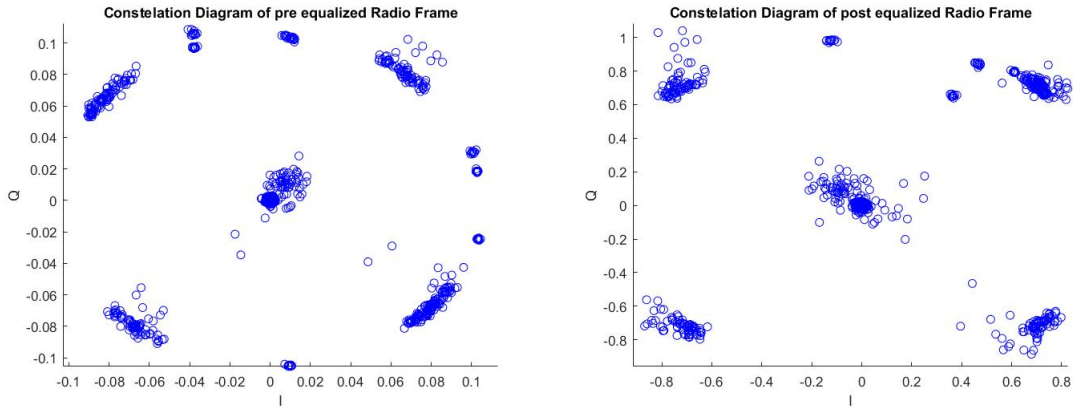


Figure 6.16: Time Symbol Offset correction. On the left before correction, on the right after correction

The occupied subframes are clearly visible, the captured samples included another radio frame's NSSS which is correctly ignored.

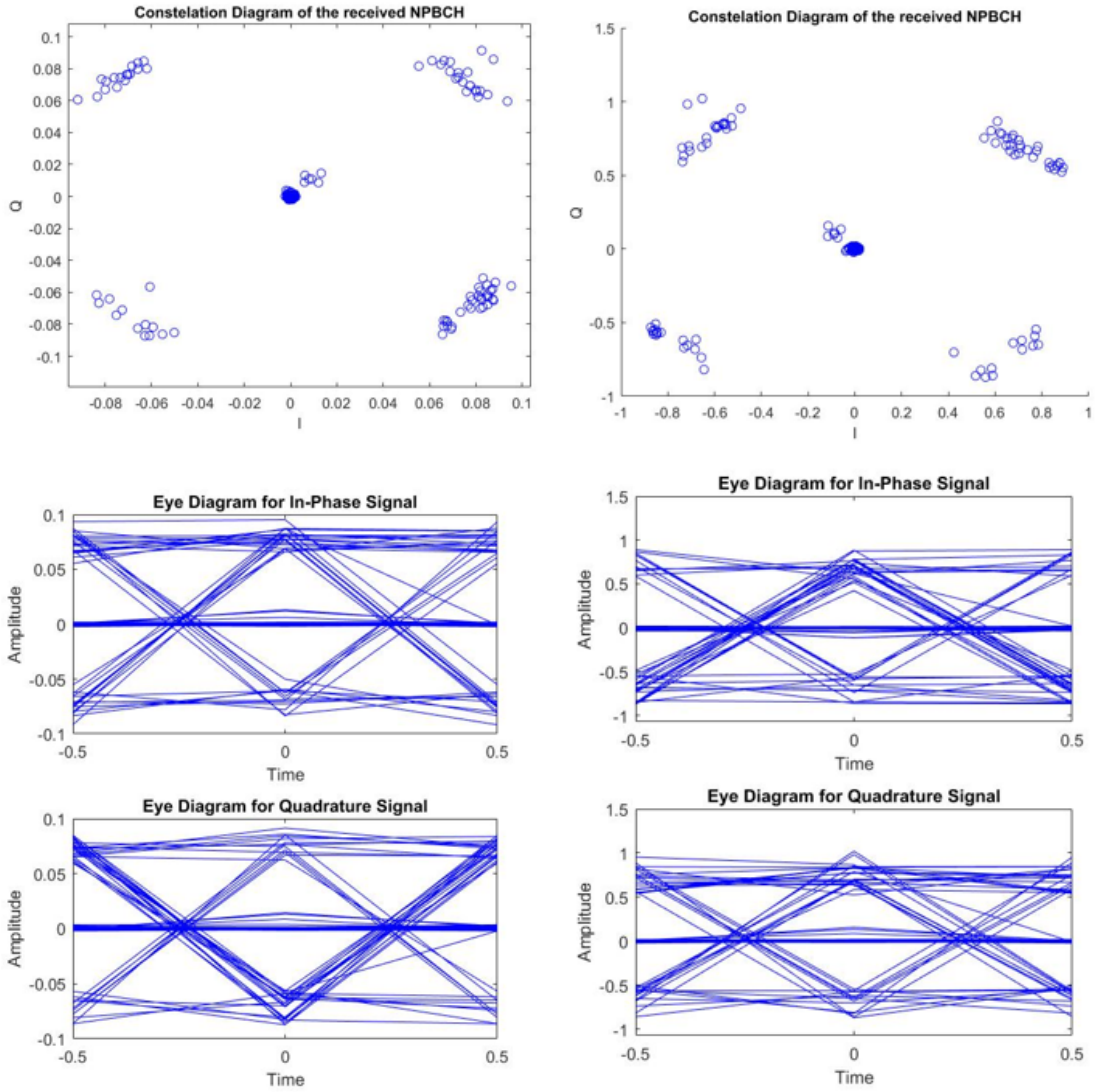


(a) Full Co-simulation received Frame constellation diagram pre equalization

(b) Full Co-simulation received Frame constellation diagram post equalization

Figure 6.17: Full received radio frame pre and post equalization

In figure 6.17 a) is possible to see that the signal is severely attenuated. In figure 6.17 b) the performance of the equalizer is proven very useful and correctly performed since it was able to reinstate the samples to the desired values noticeably verifying the QPSK symbols in their reference regions. The Zadoff-Chu sequences aren't noticeable in this case, due to the severe attenuation and since the NPSS and NSSS don't have NRS resource elements needed for the equalization process.

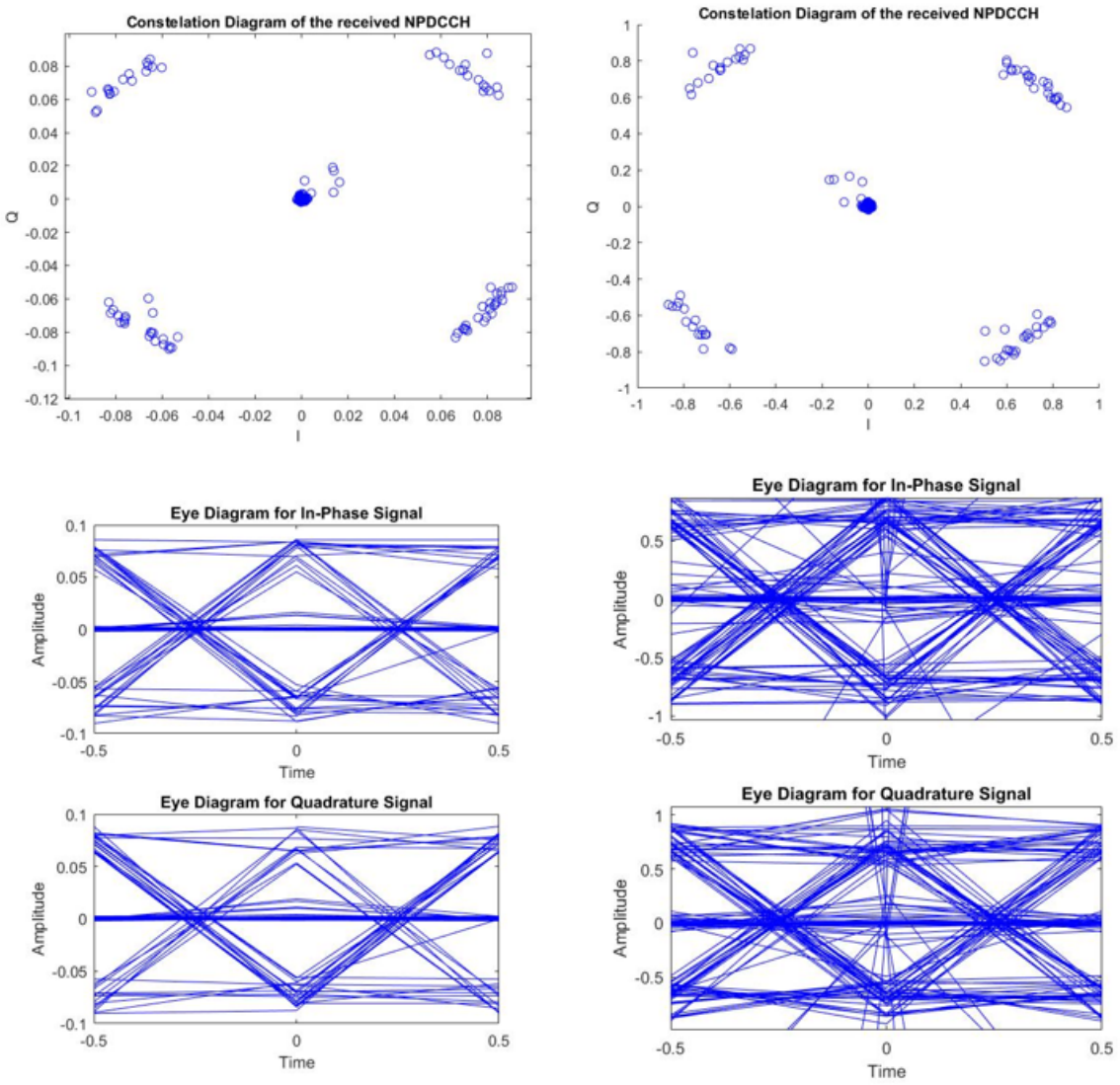


(a) Constellation and eye diagram of the received NPBCH without equalization

(b) Constellation and eye diagram of the received NPBCH with equalization

Figure 6.18: Received NPBCH subframe

The NPBCH was successfully received and the MIB correctly extracted. Figure 6.18 clearly shows a constellation diagram for QPSK symbols. The eye diagrams are reasonably open and clear which indicate a good reception. In figure 6.18 a) the signal is clearly very atenuated as can be seen by the scale of the constelation diagram. In figure 6.18 b) the equalized signal presents the desired amplitudes.

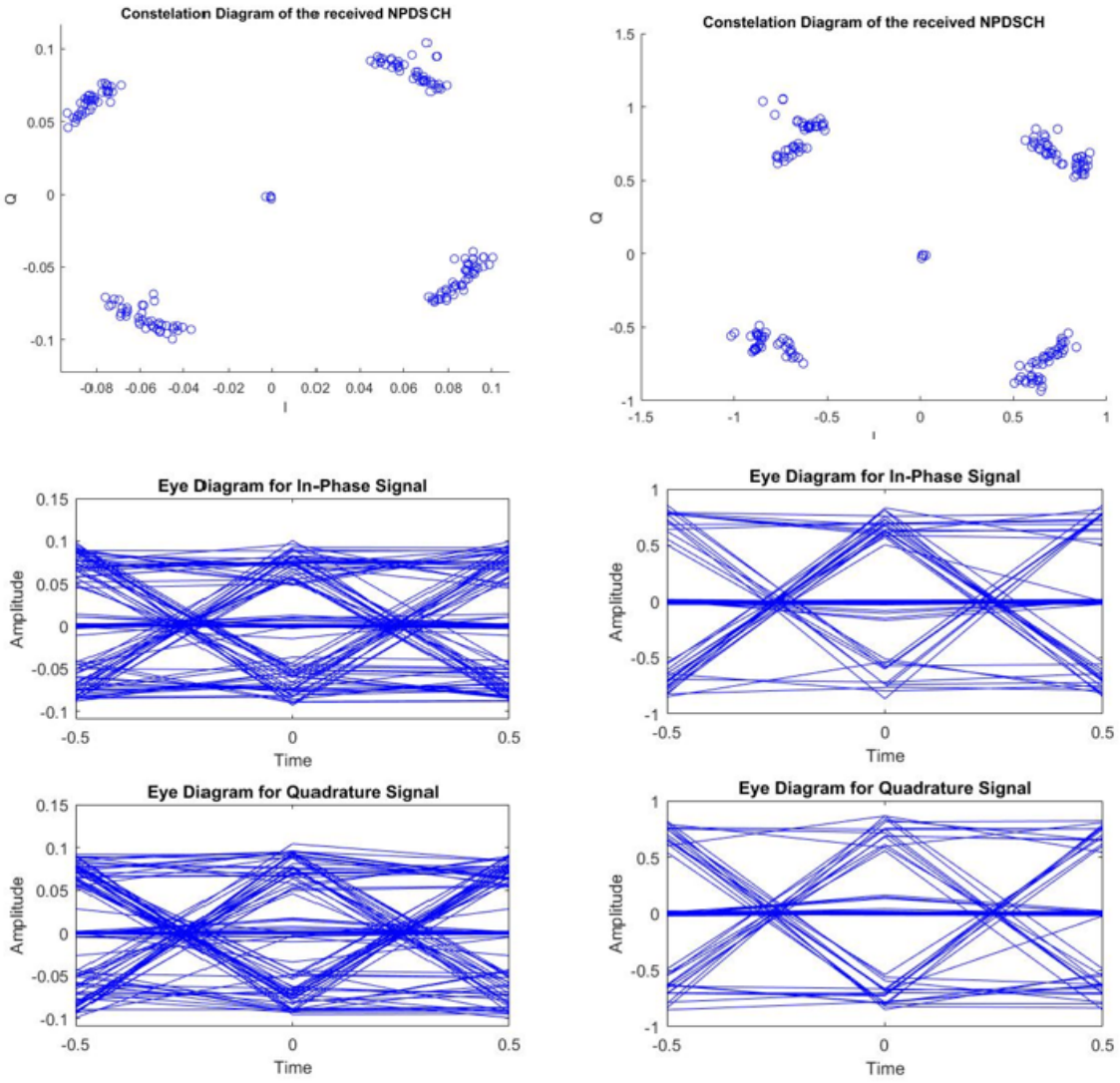


(a) Constellation and eye diagram of the received NPDCCH without equalization

(b) Constellation and eye diagram of the received NPDCCH with equalization

Figure 6.19: Received NPDCCH subframe

The NPDCCH was successfully received and the DCI message was correctly extracted. Figure 6.19 clearly shows a constellation diagram for QPSK symbols. The eye diagrams in figure 6.19 present some distortion although it's clear to identify the QPSK symbol reception. The damage presented in the eye diagram is due to the thermal noise which the Z-F equalizer does not compensate correctly. This is made really clear when comparing figures 6.19 a) and b) the atenuation in a) is severe and corrected in b) even though the additive noise is not correctly compensated.



(a) Constellation and eye diagram of the received NPDSCH without equalization

(b) Constellation and eye diagram of the received NPDSCH with equalization

Figure 6.20: Received NPDSCH subframe

The NPDSCH was also successfully received and the Shared Channel message was correctly extracted.

Figure 6.20 clearly shows a constellation diagram for QPSK symbols. The eye diagrams demonstrate a clear reception with low distortion. The eye diagram in figure 6.20 b) is fairly open and clear indicating a good reception when the signal is equalized. In figure 6.20 a) the signal is atenuated and corrected in figure 6.20 b).

6.3.1 Power Spectrum

The transmission and reception functions plot the power spectrum of the transmitted and received baseband signal, as mentioned in the above chapter. The spectrum analyzer outputs are presented in figures 6.21 and 6.22.

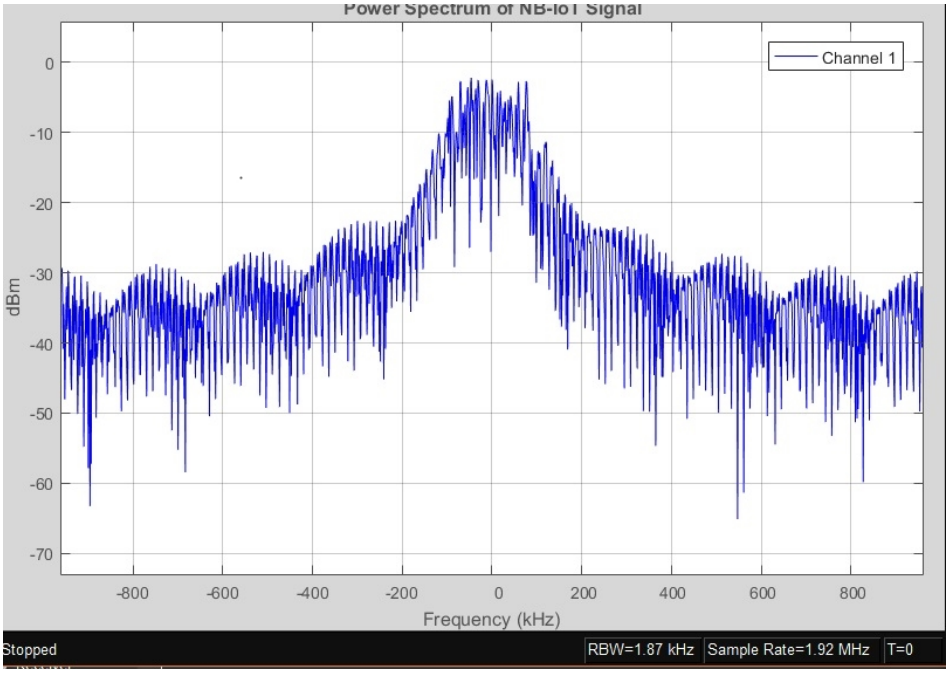


Figure 6.21: Transmitted radio frame power spectrum

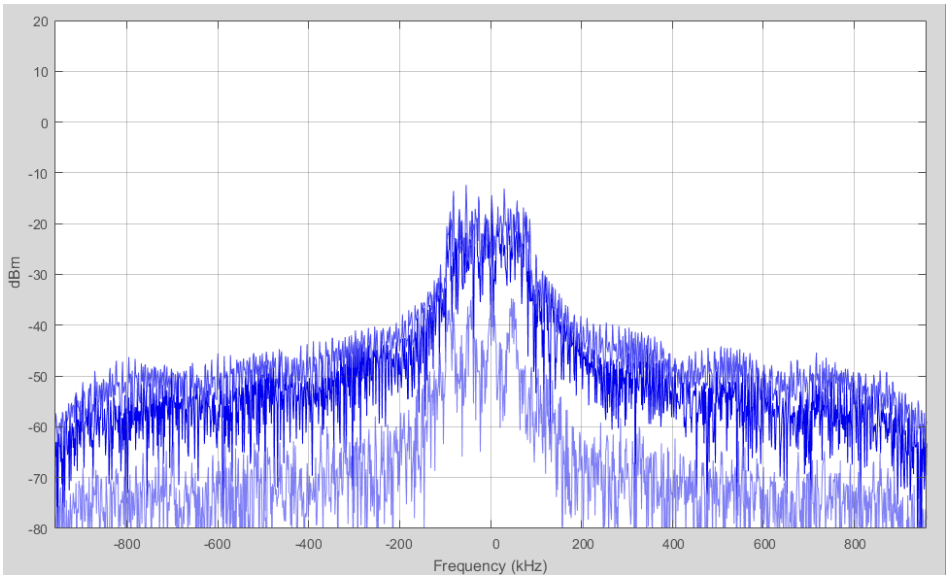


Figure 6.22: Received radio frame power spectrum

The plot of the transmission's power spectrum is issued before the gain application as, as

seen in figure 6.21, the output power isn't normalized at 0dBm. This is due to the empty subframes with many inserted zeros with lowers the average bit power. The effect can be corrected in future implementations with power control functions. The power spectrum makes clear the signal's bandwith of 180kHz which proves to be standard compliant.

The system's performance for both simulation and co-simulation environments was assessed and demonstrated positive results. In the next chapter the considerations regarding the project are present along with future work proposals.

Chapter 7

Conclusion

The proposed work of developing a behavioral dowlink model for NB-IoT was successful. The developed project is able to generate the standard's physical channels (NPDSCH, NPBCH, NPDCCH) and physical signals (NRS, NPSS, NSSS) for standalone deployment with one antenna port.

7.1 Remarks

The transmission procedures (CRC, tail-biting convolutional coding, rate matching, modulation and scrambling) are performed correctly and standard compliant. The same applies for the reception procedures which are symmetrical.

The simulation environment allows testing for various channel models. The Z-F equalizer proved efficient for compensating multipath fading and channel attenuation behaving, as expected, poorly for additive noise correction.

The integration with the SDR platform, USRP, was also successfully achieved for transmission of one radio frame. The main goal was to create a radio link for NB-IoT samples which was successful. With further testing the quality of transmission using the USRP can be improved with better filter choices.

Developing open source scripts and self-contained code, not using already existing functions and structures allows the project to be easily adapted to the developers' needs. Nevertheless, it's a time consuming choice regarding code development. With the exception of the tail-biting decoder all functions are adaptable. The decoding step was made by a Matlab toolbox function due to time constraints.

7.2 Future Work

To further improve the project and truly make it a valuable toolkit for developers, more procedures must be implemented such as cell access, cell search and scheduling, within others.The future work on this project must work towards a model for developing transceivers or NB-IoT applications.

The design of the tail-biting decoder is important in order to avoid using the one from Matlab.

Implementing the missing dynamic scheduling procedures will allow further testing to be performed.

Future work should also consider implementing the two remaining deployment modes and layer map the samples for two antenna ports.

Finally fully integrate the system with the uplink counterpart dissertation and evolve th system for integration with a C-RAN testbed.

Bibliography

[3GP16] 3GPP211. TS 136 211 - V13.2.0 -LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation (3GPP TS 36.211 version 13.2.0 Release 13. 2016.

[3GP17a] 3GPP212. TS 136 212 - V13.4.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 13.4.0 Release 13). 2017.

[3GP17b] 3GPP213. TS 136 213 - V13.3.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 13.3.0 Release 13). 2017.

[3GP17c] 3GPP331. TS 136 331 - V13.3.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol Specification (3GPP TS 36.331 version 13.3.0 Release 13). 2017.

[dSC10] Joao Carlos da Silva Carreira. *Implementacao em hardware de um analisador de espectros baseado em SDR*, 2010.

[Lab17] Link Labs. *A comprehensive look at Low Power, Wide Area Networks for Internet of things Engineers and Decision Makers*, 2017. [Online] Available at: `https://www.thethingsnetwork.org/` [Accessed in 11-2017].

[Mat] Mathworks. *lteFrequencyOffset*. [Online] Available at: `https://www.mathworks.com/help/lte/ref/ltefrequencyoffset.html` [Accessed in 11-2017].

[Mat17a] Mathworks. *LTE Downlink Channel Estimation and Equalization*, 2017. [Online] Available at: `https://www.mathworks.com/help/lte/examples/lte-downlink-channel-estimation-and-equalization.html` [Accessed in 11-2017].

[Mat17b] Mathworks. *lteconvolutionaldecode*, 2017. [Online] Available at: `https://www.mathworks.com/help/lte/ref/lteconvolutionaldecode.html` [Accessed in 11-2017].

[Mat17c] Mathworks. *lteDLFrameOffset*, 2017. [Online] Available at: `https://www.mathworks.com/help/lte/ref/ltedlframeoffset.html` [Accessed in 11-2017].

[Mat17d] MathWorks. *SDRu Receiver*, 2017. [Online] Available at: `https://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrureceiver.html` [Accessed in May 12th, 2017].

[Mat17e] MathWorks. *SDRu Transmitter*, 2017. [Online] Available at: https://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrutransmitter.html [Accessed in May 12th, 2017].

[Mau17] Robert Maunder. *Matlab UMTS/LTE Turbo Code*, 2017. [Online] Available at: http://users.ecs.soton.ac.uk/rm/resources/matlabturbo/ [Accessed in 11-2017].

[Mit95] J Mitola. The Software Radio Architecture. IEEE Communications Magazine, 33(5):2638, May 1995. 1995.

[Mon10] Teofilo Jose Marques Monteiro. Projecto de um analisador de espectros baseado em SDR. 2010.

[Net17] The Things Network. *Building a global Internet of things network together*, 2017. [Online] Available at: https://www.thethingsnetwork.org/ [Accessed in 11-2017].

[Nok16] Nokia. *LTE evolution for IoT connectivity Nokia*, 2016.

[Nut17] Nutaq. *An Introduction To Orthogonal Frequency Division Multiplex (OFDM)*, 2017. [Online] Available at: https://www.nutaq.com/blog/introduction-orthogonal-frequency-division-multiplex-ofdm [Accessed in 11-2017].

[Res17a] Ettus Research. *UHD (USRP Hardware Driver)*, 2017. [Online] Available at: https://www.ettus.com/sdr-software/detail/usrp-hardware-driver [Accessed in 11-2017].

[Res17b] Ettus Research. *USRP B200/B210 datasheet*, 2017. [Online] Available at: https://www.ettus.com/content/files/b200-b210/spec/sheet.pdf[accessed on 11-2017].

[Sem17] Semtech. *What is LoRa*, 2017. [Online] Available at: http://www.semtech.com/wireless-rf/internet-of-things/what-is-lora/ [Accessed in 11-2017].

[Sha17] ShareTechnote. *LTE handbook*, 2017. [Online] Available at: http://www.sharetechnote.com/html/Handbook/LTE/NB/LTE.html[accessed on 11-2017].

[Sig17] Sigfox. *Sigfox*, 2017. [Online] Available at: https://www.sigfox.com/en [Accessed in 11-2017].

[SR16] J Schlienz and D Raddino. Narrowband Internet of Things Whitepaper. 2016.

[Tec16] Keysight Technologies. *Internet of Things (IoT)*, 2016. poster.

[TK17] Douglas Troha and Piotr Krasowski. Wireless system design NB-IoT downlink simulator. 2017.

[Vod17] Vodafone. *Preparing Vodafone networks for multi-vendor NB-IoT deployments*, 2017. [Online] Available at: http://www.vodafone.com/content/index/what/technology-blog/multi-vendor-nbiot.html# [Accessed in 11-2017].

[Wav17a] Gaussian Waves. *Eb/N0 Vs BER for BPSK over Rayleigh Channel and AWGN Channel*, 2017. [Online] Available at: https://www.gaussianwaves.com/2011/05/ebn0-vs-ber-for-bpsk-over-rayleigh-channel-and-awgn-channel-2/ [Accessed in 11-2017].

[Wav17b] Gaussian Waves. *Preferred Pairs m-sequences generation for Gold Codes*, 2017. [Online] Available at: https://www.gaussianwaves.com/2010/10/preferred-pairs-m-sequences-generation-for-gold-codes-2/ [Accessed in 11-2017].

[Wor17] RF Wireless World. *Zadoff chu sequence in LTE*, 2017. [Online] Available at: http://www.rfwireless-world.com/Terminology/Zadoff-chu-sequence-LTE.html [Accessed in 11-2017].

[WS95] Lei Wei and Christian Schlegel. *Synchronization Requirements for Multi-user OFDM on Satellite Mobile and Two-path Rayleigh Fading Channels*, 1995.

Appendix A

User Guide

This appendix serves as a guide for using and controlling developed NB-IoT downlink Physical Layer simulator.

A.1 Simulation

The simulation environment for applying different channel models and obtaining the constellations and eye diagrams is implemented in *simulradio.m*.

The script first calls the *generateDLsignal* function. To generate the signal user input will be necessary, deciding the CellID, RNTI, DCI and NCCE format. The filling of these parameters can be done by simply pressing ENTER and using the pre-defined values.

While running *generatedDLsignal* plots for the transmitted channels will appear presenting the constellation and eye diagrams.

The channel model applied is chosen by commenting or uncommenting the models in *simulradio*. If desired, the frame correction functions can also be tested by uncommenting the *Offset* section in the script.

After channel effect application the function automatically follows to recover the transmitted data in *recoverDLsignal*. Before presenting the received data the function will present the constellation and eye diagrams for each received channel.

A.2 USRP Co-simulation

To run the co-simulation two USRP B200 or B210 boards are required and two computers. The Matlab version used must be at least Matlab 2016a and the USRP package must be installed.

The user must first run script *sdruTx*. The script will call the *generateDLsignal* and user input is required. After generating the baseband samples, defining the USRP object the transmission starts and the user receives the message:

```
Starting transmission
Please run sdruRx.m in a new MATLAB session
```

The transmission will run for 5 minutes and terminates by presenting the message:

```
Transmission finished
```

The user must meanwhile run *sdruRx* in another computer. The script will capture and recover the samples. Before presenting the recovered data the eye and constellation diagrams for the received channels are presented.

Appendix B

Example signals

This appendix presents the step by step outputs for each procedure for generating the NB-IoT downlink samples.

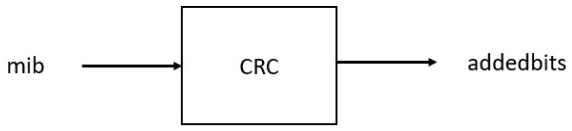
This section will present example sequences for the generation of the NPBCH, NPDCCH, NPDSCH, NRS, NPSS and NSSS.

B.1 NPBCH

The NPBCH carries the MIB which is a 34 bit word.

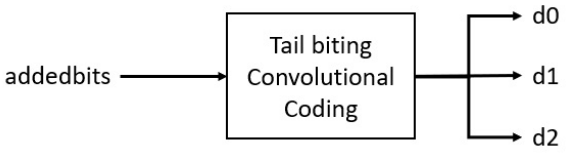
```
mib =  [1 0 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1 0 0 1]
```

First a length 16 CRC sequence is appended.




```
addedbits =  [1 0 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

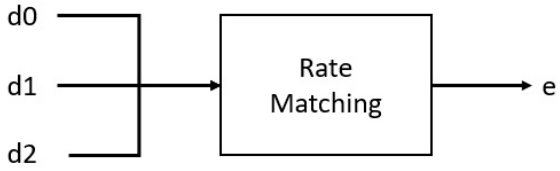
Encoding is applied.




```
d0 = [1 0 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0
      0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0]
d1 = [1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1
```

```
      0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
d2 = [1 1 0 0 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1
      1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0]
```

The codeword is permuted and matched to the payload size.

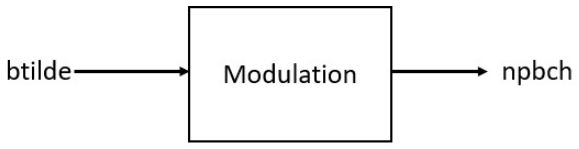



```
e = [0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 0
   0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0
   1 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1
   0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1
   1 0 1 1 1 0 0]
```

The message is encrypted and long sequences of equal values are interrupted.


```
btilde = [0 0 1 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 1
  1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1
  1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 0 1 1 1
  1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0 0
  1 1 1 0 0]
```

Finally the bits are modulated into QPSK complex-values.




```
npbch= [0.7071+0.7071i   -0.7071+0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
       -0.7071-0.7071i   -0.7071-0.7071i   0.7071-0.7071i  -0.7071-0.7071i
        0.7071+0.7071i    0.7071-0.7071i   0.7071+0.7071i  -0.7071-0.7071i
        0.7071-0.7071i   -0.7071-0.7071i   0.7071-0.7071i  -0.7071-0.7071i
       -0.7071-0.7071i   -0.7071+0.7071i   0.7071+0.7071i   0.7071-0.7071i
```

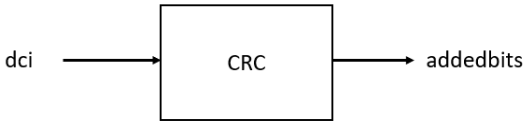
```
-0.7071-0.7071i   0.7071+0.7071i  -0.7071-0.7071i  -0.7071+0.7071i
-0.7071+0.7071i   0.7071+0.7071i   0.7071+0.7071i  -0.7071+0.7071i
-0.7071+0.7071i   0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i
-0.7071-0.7071i  -0.7071-0.7071i  -0.7071+0.7071i  -0.7071-0.7071i
-0.7071-0.7071i  -0.7071+0.7071i   0.7071-0.7071i   0.7071+0.7071i
 0.7071+0.7071i   0.7071+0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
 0.7071-0.7071i  -0.7071+0.7071i  -0.7071+0.7071i   0.7071+0.7071i
-0.7071+0.7071i  -0.7071+0.7071i   0.7071-0.7071i  -0.7071-0.7071i
 0.7071-0.7071i  -0.7071-0.7071i  -0.7071+0.7071i  -0.7071-0.7071i
 0.7071+0.7071i  -0.7071+0.7071i  -0.7071-0.7071i  -0.7071-0.7071i
 0.7071+0.7071i   0.7071+0.7071i   0.7071+0.7071i   0.7071+0.7071i
 0.7071-0.7071i  -0.7071+0.7071i  -0.7071+0.7071i  -0.7071-0.7071i
-0.7071+0.7071i   0.7071-0.7071i  -0.7071+0.7071i   0.7071+0.7071i
 0.7071-0.7071i  -0.7071-0.7071i   0.7071+0.7071i]
```

B.2 NPDCCH

The NPDCCH carries the control information. In this example a DCI N1 message is sent to schedule the NPDSCH with a 16 bits TBS.

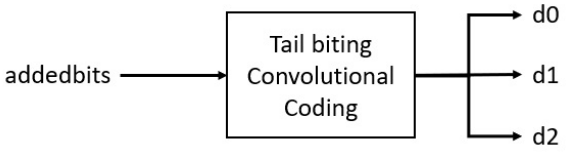
```
dci =  [ 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0]
```

First, a length 24 CRC is appended.



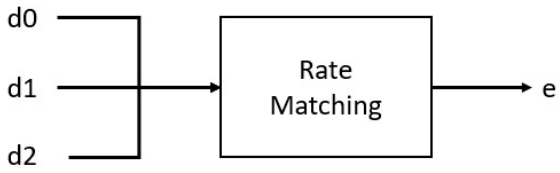

```
addedbits =  [ 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 1 1
     0 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1]
```

Encoding is applied.




```
d0 = [1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 0
      1 0 1 0 0 1 0 0 0 0 1 1 0]
d1 = [0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 1 0 0 1 1 0
      0 1 0 0 1 0 0 1 0 0 1 1 0]
d2 = [0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1
      1 1 1 1 0 1 0 0 0 0 0 0 0]
```

The codeword is permuted and matched to the payload size.




```
e = [1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
     1 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1
     0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
     1 1 0 1 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1
     0 1]
```

The message is encrypted and long sequences of equal values are interrupted.


```
btilde = [1 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0
    0 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 0
    1 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0
    1 1 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1
    1 0 0 1 1]
```

Finally the bits are modulated into QPSK complex-values.


```
npdcch= [ -0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i   0.7071+0.7071i
          -0.7071-0.7071i   0.7071+0.7071i  -0.7071+0.7071i   0.7071-0.7071i
           0.7071+0.7071i  -0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i
           0.7071+0.7071i   0.7071-0.7071i   0.7071+0.7071i   0.7071+0.7071i
           0.7071+0.7071i  -0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i
          -0.7071+0.7071i  -0.7071-0.7071i  -0.7071-0.7071i  -0.7071+0.7071i
           0.7071+0.7071i   0.7071-0.7071i  -0.7071-0.7071i  -0.7071-0.7071i
          -0.7071-0.7071i   0.7071+0.7071i   0.7071-0.7071i  -0.7071+0.7071i
```

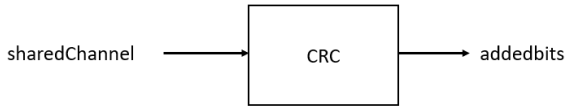
```
-0.7071-0.7071i   0.7071-0.7071i   0.7071-0.7071i  -0.7071-0.7071i
 0.7071-0.7071i  -0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i
 0.7071+0.7071i  -0.7071-0.7071i  -0.7071+0.7071i   0.7071+0.7071i
 0.7071-0.7071i   0.7071+0.7071i  -0.7071-0.7071i   0.7071+0.7071i
 0.7071-0.7071i  -0.7071+0.7071i   0.7071+0.7071i  -0.7071-0.7071i
 0.7071+0.7071i  -0.7071-0.7071i  -0.7071+0.7071i   0.7071-0.7071i
-0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i   0.7071+0.7071i
 0.7071-0.7071i  -0.7071+0.7071i   0.7071-0.7071i  -0.7071-0.7071i
 0.7071+0.7071i   0.7071-0.7071i  -0.7071+0.7071i  -0.7071-0.7071i
-0.7071-0.7071i   0.7071+0.7071i  -0.7071-0.7071i]
```

B.3 NPDSCH

The generated NPDSCH carries randomly generated data of size TBS.

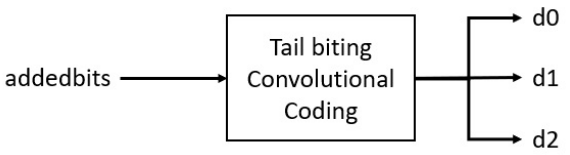
```
sharedChannel =  [ 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0]
```

First a length 24 CRC sequence is appended.



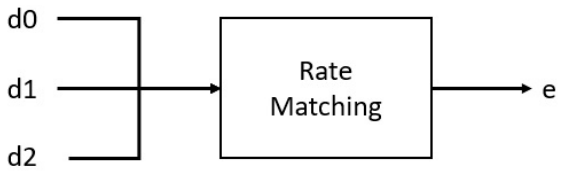

```
addedbits =  [1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1
      1 0 1 0 0 0 0 1 0 0]
```

Encoding is applied.




```
d0 = [0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1
      0 1 1 0 1 1]
d1 = [0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 0
      1 0 1 1 0 1]
d2 = [1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0
      0 1 0 1 0 1]
```

The codeword is permuted and matched to the payload size.




```
e = [0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 1
     0 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0
     1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1
     1 1 0 1 1 0 1 0 1 1 0 1]
```

The message is encrypted and long sequences of equal values are interrupted.


```
btilde = [ 0 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0
           0 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0
           0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0
           0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1]
```

Finally, the bits are modulated into QPSK complex-values.

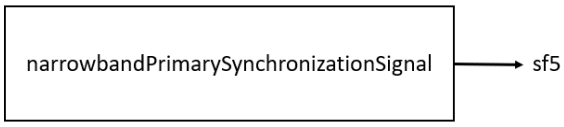

```
npdsch= [  0.7071+0.7071i   0.7071-0.7071i  -0.7071+0.7071i  -0.7071-0.7071i
           0.7071+0.7071i   0.7071+0.7071i   0.7071-0.7071i   0.7071+0.7071i
           0.7071+0.7071i   0.7071+0.7071i   0.7071-0.7071i  -0.7071+0.7071i
           0.7071+0.7071i  -0.7071+0.7071i  -0.7071-0.7071i   0.7071-0.7071i
           0.7071+0.7071i  -0.7071-0.7071i   0.7071-0.7071i  -0.7071-0.7071i
           0.7071-0.7071i   0.7071-0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
          -0.7071+0.7071i   0.7071+0.7071i  -0.7071+0.7071i   0.7071+0.7071i
          -0.7071+0.7071i  -0.7071-0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
          -0.7071+0.7071i   0.7071-0.7071i   0.7071-0.7071i  -0.7071-0.7071i
           0.7071+0.7071i  -0.7071-0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
          -0.7071-0.7071i   0.7071-0.7071i   0.7071-0.7071i   0.7071+0.7071i
           0.7071-0.7071i  -0.7071+0.7071i   0.7071-0.7071i   0.7071+0.7071i
          -0.7071-0.7071i   0.7071+0.7071i   0.7071-0.7071i   0.7071+0.7071i
```

```
   0.7071+0.7071i   0.7071+0.7071i  -0.7071+0.7071i  -0.7071+0.7071i
  -0.7071+0.7071i   0.7071-0.7071i   0.7071-0.7071i   0.7071-0.7071i]
```

B.4 NRS


```
map=
0   0   0   0   0   0.7071+0.7071i   0
0   0   0   0   0   0                0
0   0   0   0   0   0                0
0   0   0   0   0   0                0.7071+0.7071i
0   0   0   0   0   0                0
0   0   0   0   0   0                0
0   0   0   0   0   0.7071+0.7071i   0
0   0   0   0   0   0                0
0   0   0   0   0   0                0
0   0   0   0   0   0                0.7071+0.7071i
0   0   0   0   0   0                0
0   0   0   0   0   0                0
```

B.5 NPSS



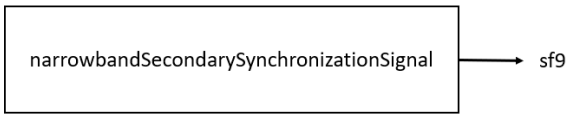

```
sf5=
Columns 1 through 8
0   0   0  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i
0   0   0  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i
0   0   0  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i
0   0   0  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i
0   0   0   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i
0   0   0  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i
0   0   0  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i
0   0   0  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i
0   0   0  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i
0   0   0   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i
```

```
0   0   0   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i
0   0   0   0    0   0     0                0

Columns 9 through 14

-0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i
-0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i
-0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i
-0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i
 0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i   0.4154+0.9096i
-0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i  -0.9595+0.2817i
-0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i  -0.1423+0.9898i
-0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i  -0.6549-0.7557i
-0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i  -0.9595-0.2817i
 1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i
 1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i   1.0000-0.0000i
       0                0                0                0                0                0
```

B.6 NSSS




```
sf9=
Columns 1 through 8

0   0   0   0.5593-0.8290i   0.8610+0.5086i   0.9849+0.1730i   0.9921+0.1256i  -0.4668-0.8844i
0   0   0   0.8926-0.4508i  -0.9317+0.3632i  -0.5443-0.8389i   0.3067+0.9518i  -0.8816-0.4721i
0   0   0   0.9948-0.1017i   0.1493-0.9888i  -0.1017+0.9948i  -0.6268+0.7792i  -0.9699+0.2433i
0   0   0   0.9928+0.1196i   0.8640+0.5035i   0.5936-0.8048i  -0.9988+0.0479i  -0.4292+0.9032i
0   0   0   0.9805+0.1966i  -0.6899+0.7239i  -0.8548+0.5189i  -0.7940-0.6080i   0.5493+0.8356i
0   0   0   0.9913+0.1315i  -0.6590-0.7522i   0.9554-0.2953i  -0.3520-0.9360i   0.9839-0.1789i
0   0   0   0.9970-0.0779i   0.7155-0.6986i  -0.9805+0.1966i   0.0539-0.9985i   0.0779-0.9970i
0   0   0   0.9083-0.4184i   0.8256+0.5643i   0.9714-0.2375i   0.3181-0.9480i  -0.9817-0.1907i
0   0   0   0.5984-0.8012i  -0.2550+0.9670i  -0.9108+0.4129i   0.4347-0.9006i  -0.1612+0.9869i
0   0   0   0.0000-1.0000i  -0.9742+0.2259i   0.7280-0.6856i   0.4184-0.9083i   0.9999-0.0120i
0   0   0  -0.7071-0.7071i  -0.7561-0.6544i  -0.3351+0.9422i   0.2665-0.9638i  -0.3238-0.9461i
0   0   0  -0.9897+0.1434i  -0.0599-0.9982i  -0.2723-0.9622i  -0.0360-0.9994i  -0.7029+0.7113i

Columns 9 through 14

 0.9057+0.4238i   0.4347-0.9006i  -0.1612+0.9869i  -0.9108+0.4129i  -0.2550+0.9670i   0.5984-0.8012i
-0.9879-0.1553i  -0.9480-0.3181i  -0.1907+0.9817i  -0.2375-0.9714i   0.5643-0.8256i  -0.4184-0.9083i
 0.9638-0.2665i  -0.0539+0.9985i  -0.0779+0.9970i   0.9805-0.1966i  -0.7155+0.6986i  -0.9970+0.0779i
-0.6679+0.7442i   0.9360-0.3520i   0.1789+0.9839i   0.2953+0.9554i   0.7522-0.6590i  -0.1315+0.9913i
 0.0180-0.9998i  -0.7940-0.6080i   0.5493+0.8356i  -0.8548+0.5189i  -0.6899+0.7239i   0.9805+0.1966i
 0.7442+0.6679i   0.0479+0.9988i   0.9032+0.4292i  -0.8048-0.5936i   0.5035-0.8640i   0.1196-0.9928i
-0.9638+0.2665i   0.6268-0.7792i   0.9699-0.2433i   0.1017-0.9948i  -0.1493+0.9888i  -0.9948+0.1017i
 0.1553-0.9879i  -0.9518+0.3067i   0.4721-0.8816i   0.8389-0.5443i  -0.3632-0.9317i   0.4508+0.8926i
 0.9057+0.4238i   0.9921+0.1256i  -0.4668-0.8844i   0.9849+0.1730i   0.8610+0.5086i   0.5593-0.8290i
-0.5443+0.8389i  -0.9132-0.4075i  -0.9994+0.0360i   0.7113+0.7029i  -0.9622+0.2723i  -0.9982+0.0599i
-0.8421-0.5393i   0.8421+0.5393i  -0.2665+0.9638i   0.3238+0.9461i   0.3351-0.9422i   0.7561+0.6544i
 0.4075-0.9132i  -0.8389-0.5443i   0.9083+0.4184i   0.0120+0.9999i   0.6856+0.7280i  -0.2259-0.9742i
```